8-2016

Detection of communication over DNSSEC covert channels

Nicole M. Hands
Purdue University

Graduate School Form
30 Updated 12/26/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Nicole M Hands

Entitled

Detection of Communication Over DNSSEC Covert Channels

For the degree of Master of Science

Is approved by the final examining committee:

Baijian Yang
Chair

Marcus Rogers
Co-chair

Dongyan Xu
Co-chair

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Baijian Yang

Approved by: Jeffrey Whitten 7/21/2016

Head of the Departmental Graduate Program Date

DETECTION OF COMMUNICATION OVER DNSSEC COVERT CHANNELS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Nicole M. Hands

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

August 2016

Purdue University

West Lafayette, Indiana

ACKNOWLEDGMENTS

First, this work was funded, in part, by National Science Foundation grant Award 1027493, and the author is grateful for that support.

The author would like to express sincere appreciation to the members of the Master's thesis committee who gave their time, energy, and knowledge in support of this project. This work is based on an earlier work: A Study on Botnets Utilizing DNS, in Proceedings of the 4th Annual ACM Conference on Research in Information Technology, (2015, September) Âľ ACM, 2015. http://dx.doi.org/10.1145/2808062.2808070

The author would also like to thank Professor Raymond Hansen and Dr. Baijan Yang for their contributions to the precursor paper on which this study was built.

Further, the support of Dr. Filipo Shrevski of Purdue University; Dr. Roland van Rijswijk-Deij of University of Twente, Netherlands; and the volunteers who operate Passive DNS sensors must be acknowledged wholeheartedly for their guidance and assistance in locating data and in problem solving.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

BIND	Berkeley Internet Name Domain
CAIDA	Center for Applied Internet Data Analysis
C&C	Command and Control
CDN	Content Delivery Network
DDoS	Distributed Denial of Service
DIG	Domain Information Groper
DNS	Domain Name Service
DNSSEC	Domain Name Service Security Extension
DoS	Denial of Service
EDNS0	Extended DNS, version 0
FN	False Negative
FP	False Positive
HTTP	Hypertext Transport Protocol
ICT	Information & Communications Technology
IDS	Intrusion Detection System
IP	Internet Protocol
IRC	Internet Relay Chat
ITAP	Information Technology at Purdue
P2P	Peer to Peer
PII	Personally Identifiable Information
QID	Query Identificator
RC4	Rivest Cipher 4, also known as Ron's Code
RPZ	Response Policy Zone
SEIM	Security Event and Incident Management

SETI	Search for ExtraTerrestrial Intelligence
SOA	Start of Authority
SURF	Samenwerkende Universitaire Reken Faciliteiten
TCP	Transport Control Protocol
TN	True Negative
TP	True Positive
UDP	User Datagram Protocol

GLOSSARY

Bailiwick	The set of domains and subdomains over which a given DNS server has authority to maintain records.
Botnet	A collection of computers running a software program to carry out the functions of a remote and user. These can have both legitimate purposes as well as malicious ones.
Botmaster	The individual or individuals sending commands remotely to the infected machines in the botnet. Can also be referred to as a bot herder.
Covert Channel	The study will use the definition put forth by Tsai, Gligor, and Chandersekaran (1990) of a covert channel. It is as follows:

> "given a nondiscretionary security model M and its interpretation $I(M)$ in an operating system, any potential communication between two subjects $I(S_i)$ and $I(S_j)$ of $I(M)$ is covert if and only if any communication between the corresponding subjects $I(S_i)$ and $I(S_j)$ of the model M is illegal in M" (p.1).

C&C	The computer or computers in control of the botnet's actions
Malware	A portmanteau created from the words "Malicious" & "Software"

ABSTRACT

Hands, Nicole M. M.S., Purdue University, August 2016. Detection of Communication Over DNSSEC Covert Channels. Major Professors: Baijian Yang, Marcus Rogers, Dongyan Xu.

Unauthorized data removal and modification from information systems represents a major and formidable threat in modern computing. Security researchers are engaged in a constant and escalating battle with the writers of malware and other methods of network intrusion to detect and mitigate this threat. Advanced malware behaviors include encryption of communications between the server and infected client machines as well as various strategies for resilience and obfuscation of infrastructure. These techniques evolve to use any and all available mechanisms. As the Internet has grown, DNS has been expanded and has been given security updates. This study analyzed the potential uses of DNSSEC as a covert channel by malware writers and operators. The study found that changing information regarding the Start of Authority (SOA) and resigning the zone can create a covert channel. The study provided a proof of concept for this previously undocumented covert channel that uses DNSSEC.

CHAPTER 1. INTRODUCTION

The tone of collegiality and trust upon which the Internet was initially built has evolved into a frantic cat and mouse game of security versus criminal activity. The threat is pervasive, it is diverse, and it is ever present. The Internet never sleeps nor do the criminals who misuse it for various purposes, it seems. It is within the context of this computing environment of unyielding threat that the following work has been carried out. Data exfiltration, either of classified data or personally identifiable information, has been identified as a severe threat for many years and continues to grow in pervasiveness. Data infiltration also represents a threat to trust in information systems as well.

Covert channels have been used by cyber criminals to carry out the removal and placement of data and/or programs (such as malware). Due to the hidden nature of these communication paths, their existence may be unknown and the unsuspecting target does not even know to look for such a threat. As an example, botnets use differing methods of communicating with each other and their command and control. The Domain Name Services (DNS) upon whose functionality nearly every Internet transaction relies is used by some botnets as a covert channel.

DNSSEC was introduced in order to provide security to this foundational protocol. Like the rest of the Internet, DNS was initially built without security in mind. The a posteriori security measures of DNSSEC have attempted to remediate the known dangers to DNS (such as cache spoofing). To the best of our knowledge, at the time of writing, the use of DNSSEC as a covert channel had not yet been investigated and thus it was unknown whether or not the security extensions provided any protection against the covert pernicious form of botnet communication.

This study engaged in an attempt to investigate DNSSEC covert channels. Covert channel formation over DNSSEC by botnets and other malware represents a

significant threat to sensitive digital information, and therefore identifying covert channels before sensitive data is lost or modified is essential.

1.1 Scope

At the time of writing and to the best available knowledge, no known malware creates covert channels specifically using the security extensions to DNS (DNSSEC). This study attempted to ascertain what potential exists for exploitation of the security extensions and the mechanisms associated with updating zone information and signatures. Specifically, the study asks does a unique covert channel exist within DNSSEC that would not be identified by Intrusion Detection Systems and Security Information and Event Management (SIEM) systems in their present configurations?

For the purposes of this study, a protocol analysis was carried out and when a potential weakness was identified, a proof of concept prototype was created to demonstrate the functionality or lack of functionality of the channel. Though the protocol analysis was extensive, every potential covert channel was not investigated to the point of prototyping. These were left for future work.

1.2 Significance

Network administrators adopt tools to protect the resources with which they are entrusted. These tools constantly change as a result of the evolution of the threats against which they are tasked to defend. Any attack against the DNS infrastructure has the potential to do great harm to the millions of users who rely on it to work invisibly in the background. Attacks have successfully disrupted the Internet activities of millions of individuals, significantly in an incident in China that took place in 2014 where millions of individuals were redirected to a website that would otherwise have been blocked (Mozur, 2014).

When malware uses the DNS infrastructure, it takes advantage of a largely unmonitored and unfiltered protocol to conduct its malicious business (Fulton, 2015). Though malware can misuse DNS in a number of ways, one dangerous misuse is the creation of a covert channel to either exfiltrate or infiltrate data into or out of an unauthorized system. Consider an example corporation that owns valuable machinery and has physical assets that is worth tens of billions of dollars, while a social media site is worth hundreds of billions of dollars. This situation hints at opportunities for income generation through the theft, poisoning, or destruction of data. In fact, the market for data has many offerings, from Personally Identifiable Information (PII) to browsing history, medical records, passport scans, among many others. The prices of these vary from a dollar to hundreds of dollars according to a report from TrendMicro (Huq, 2015). With the number of records for sale in the millions, at almost any price, the potential for profit is present. Though specifics have not been reported, it is through an extension of this market driven crime model that the potential for profit could come from the destruction or modification of data as well.

Distributed computing has many uses including the processing of scientific data in projects such as SETI@home. SETI would be considered a non-malicious botnet and other scientific distributed computing systems can accomplish incredible data processing. When malware infects hundreds, thousands, or millions of computers, the processing tasks that can be accomplished are limited only by the malware writer's imagination and ability. The abilities of malware writers are moving away from those that have previously been characterized as skids or script kiddies (those who use a template to write unsophisticated code to carry out simple malicious tasks), to state level actors and organized crime. This means that researchers are increasingly needing to recognize that their target is not only moving, but accelerating. The creation of DNSSEC was meant to help address the initial unsecured infrastructure and to protect against certain specific attacks against DNS. This study is born from the recognition that, though implementation

of the DNS security extensions has been slow, that it is a new territory that the given the opportunity, cyber criminals will inhabit first, creating a situation where new adopters are not expecting that the system is already insecure. Research has been conducted to determine DNSSEC's susceptibility to attacks such as amplification attacks (van Rijswijk-Deij, Sperotto, & Pras, 2014) that also affected DNS before the extensions were added. This work attempts to investigate another aspect of DNSSEC that may be vulnerable to attack, that is it's potential for usage as a covert channel. In the present computing environment, not only is there economic risk associated with failure to secure DNSSEC against this method of communication, but these data attacks have also have political, personal, physical security, and privacy implications.

1.3 Research Question

At the time of the thesis proposal, this research proposed to seek an answer to the following research question: Given large scale DNS and DNSSEC passive traffic captures collected from global domains across TLDs, do character frequency analysis and entropy analysis measures used to detect DNS covert channels continue to uncover covert channels when DNSSEC is used? This question has been modified. This study sought instead to answer the following research question: What features of DNSSEC could be exploited to create DNSSEC specific covert channels? What characteristics would such a covert channel have?

The study investigated the hypothesis that DNSSEC specific covert channels exist.

1.4 Assumptions

This study was conducted acknowledging the following assumptions:

- A model of attacker was assumed to have remote or local access to the compromised machine.
- This attacker may be a malicious insider or a malicious stranger.
- The goal of the attacker's activity was theft or insertion of data (that is to say that the data transfer may take place from client to server or vice versa).
- The attacker was assumed to have access to (or skills to create) a tunneling tool that is sophisticated in nature, using all known present day obfuscation techniques.
- The attacker was assumed to have access to the authoritative name server for a domain. This domain could either belong to the attacker or could be hijacked.

1.5 Limitations

The limitations for this study include the following:

- The study was conducted using a registered domain name under the control of the author, however the domain was not configured as part of a chain of trust through the registrar (parent) due to an unresolved technical problem on the registrar's end. This limited the applicability of the results to islands of trust as opposed to more common chains of trust.
- DNSSEC Look-aside Validation (DLV) was no longer available for zones that could validate to the Root. As ".org" is a zone that has root signing availability, DLV was unavailable as an alternative chain of trust.

1.6 Delimitations

The delimitations for this study include:

- Though tunneling applications exist in open source format, modifying such a tool to include the new tunneling technique would not have been a trivial task. Including the full suite of functionality associated with such tools is out of scope of this work.

- The study implemented a tunnel in a direct plain-text to binary conversion without encryption. Though encryption could add to the ability of the data to remain hidden should the covert channel be discovered, it does not add to the covertness of the channel. Therefore, encryption was not implemented.

- The DNS name servers were set up using BIND version 9, due to BIND being the most common name server software, and an open source software. Bind is maintained and developed by the Internet Systems Consortium (ISC).

- The implementation was only tested using Ubuntu clients and servers. No Windows machines were tested. Windows does implement DNS differently than BIND and therefore could possibly yield different results. This research is left for future study.

1.7 Summary

This chapter has highlighted the scope, limitations, delimitations, and assumptions upon which this study was conducted. Chapter 2 reviews the relevant literature on DNS, botnets, DNSSEC and attempts to identify gaps in the literature.

CHAPTER 2. REVIEW OF RELEVANT LITERATURE

The DNS protocol is a foundational part of the Internet. It was created in RFC 1034 (Mockapetris, 1983) and RFC 1035 (Mockapetris, 1987). DNS is a distributed database of records associating IP addresses with domain names. The database also keeps track of the servers that have authority over the records associated with a given domain. Without this database, instead of being able to type in www.example.com, which is easy for a human to remember, the user would need to remember the IP address 93.184.216.34. In some cases, many websites share the same IP address, or have many IP addresses associated with them. DNS allows differentiation of these domains and makes the use of the Internet as it is known possible.

At a very basic level, the way a request by a user for a web address gets translated into a connection starts with an application, such as a web browser, asking the client-side resolver to initiate the search for the IP answer. The resolver has some data locally, including the IP address of the name server it is expected to send its query to. This name server is a recursive server. The recursive server will request data from one of the root servers who will in turn direct the recursive servers to the authoritative name server for the domain originally requested. The recursive server gives the answer it has received to the resolver, and the resolver can now establish a connection with the website's IP. Each DNS request contains different fields, including an identification number (QID), the source IP, the destination IP, source and destination ports, checksums, and the question or answer data itself. The structure of a DNS packet is discussed at length in a later section. ***

Security researchers have recognized the need for protection to DNS since at least 1995 when Paul Vixie analyzed the software BIND, and found serious flaws in the implementation of DNS through BIND (Vixie, 1995). The public was made

aware of the threats outlined by Vixie in 1997, when a protester named Eugene Kashpureff (Wayner, 1997) conducted a modification to the DNS database that directed individuals attempting to reach InterNIC's website to his own. This was possible due to the way that DNS servers cache results of queries. Kashpureff's attack was a cache poisoning attack (a type of spoofing attack), such that the records served by the resovers were the poisoned records and not the original and legitimate ones. This attack can be prevented by ensuring that forwarding DNS name servers handle only internal DNS queries. When they handle recursive external queries, they are potentially open to the cache poisoning attack. Dan Kaspersky combined the cache poisoning technique with that of correctly "guessing" the QID for a query to demonstrate another very severe vulnerability in DNS (and an effective way to spoof an answer to a DNS query).

In the information security field, three goals are traditionally identified: integrity, availability, and confidentiality. In 2002, DNS root servers were subjected to a DDoS attack, and 9 of the 13 root servers in operation at the time were affected (Baranowski, 2003). DoS and DDoS, cache poisoning or other spoofing attacks affect the availability of DNS. Cache poisoning and spoofing also affect integrity. DNS is, by nature, public information and is not generally expected to be confidential.

The malware threat landscape is constantly changing and evolving rapidly as the perpetual threat-mitigation-threat-escalation cycle continues with security researchers and malware writers both utilizing all available resources to meet their goals. Of all the modern threats, botnets are frequently cited as high priority (Comey, 2015; Norton, 2007; Thakar, 2013) due to the diverse potential for criminal activity they provide, with the number of individuals who could fall victim ranging from the hundreds in a small botnet to millions in a larger one. A botnet can be defined as a system of computers that are under the control of a criminal, typically referred to as the botmaster, after having been infected by a piece of malware (a worm, virus, Trojan) that took advantage of a vulnerability or exploit to

provide remote access and control of the machine. Though the first botnets were not malicious in nature, this rapidly changed (Tyagi & Aghila, 2011). Malicious and nefarious activities that currently involve botnets include the sending of spam (unsolicited email), distributed denial of service (DDoS) attacks, identity/password theft, remote storage of illegal files, financial theft, spread of additional malware payloads, the manipulation of online polls or games (S. S. Silva, Silva, Pinto, & Salles, 2013), among others.

With the severity of this threat in mind, the study examines the functionality of botnets and in particular it looks at the communication strategies utilized. Covert channels and encryption have been used by malware writers to help hide their malicious activities. This provides a useful context for examining covert channel functionality.

2.1 The Structure of a DNS Packet

In order to best examine the ways in which DNS can be exploited to create covert channels, the structure of the packet should be considered.

bits 0***4***8***12***16***20***24***28***32	
Query ID (QID)	Flags& Codes
Question Count (QC)	AN(swer)Count
NameServer Count	Additional Record Count
Question Name (DNS notation)	
Question Type	Question Class

Figure 2.1.: DNS Header Format

According to RFC 1035 (Mockapetris, 1987), the first 16 bit word in the DNS packet header contains a query ID number. RFC 1035 (Mockapetris, 1987) and RFC 4035 (Arends, Austein, Larson, Massey, & Rose, 2005b) define the second 16-bit word in the header as containing various flags and codes, which include the following. The "QR" flag consists of one bit to indicate message is either a query or a response. The Opcode is 4 bits, to indicate if the query is a query, status request, a notification of zone changes, or a dynamic update. The "AA" flag indicates if the answer is authoritative. The "TC" bit is set when the message is larger than a given packet size (traditionally 512 bytes) and had to be truncated. If recursion is desired, the "RD" bit is set. If recursion is available, the "RA" bit is set. Z was once 3 bits whose values were long reserved and must have been set to zero. The updates to DNS redefined thes bits. The first of which (bit 9 in the header) remains reserved. The other two bits have been allocated now for DNSSEC purposes. If the "AD" bit is set, the data is Authenticated Data (as part of DNSSEC). If "CD" is set, then checking the authentication has been disabled. The last bit of the flags and codes section is the Rcode, 4 bits to indicate server status upon replying to queries. These codes include codes for "no error", "format error", "NX RR" for a resource record set that should exist but does not or "YX RR" for a resource record set that should not exist but does.

The DNS message portion of the packet has a structure of three main sections: the question section, the authority section, and the additional section. Not all sections are present in all queries or responses.

The Question Name is formated in DNS name notation. This label is formatted to encode the fully qualified domain name, starting with the lowest level domain (LLD) and ending with the root (.). The domain "domain.example.com." would be encoded starting with two bits to indicate the type of label. Two 0's would indicate it is a standard name label. The example domain name would be encoded as 00[6]domain[7]example[3]com[0]. Each value in the square brackets indicates the length of the label to follow. The maximum length of any label is 63 bits. The

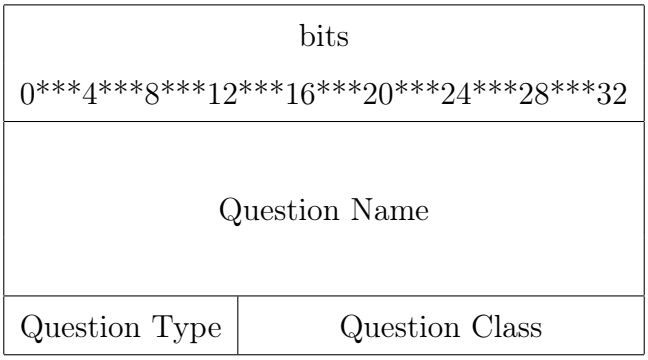

Figure 2.2.: Question Format for DNS Message

maximum total length of the domain is 255 bits. The encoding ends with the zero because the root has zero length.

Two 1's at the beginning of the label encoding indicate the label is compressed. This tells the DNS software that a pointer follows and it will then continue parsing the message from the location indicated by the pointer. The pointer exists as an offset from the beginning of the packet, starting at zero. The initial bits for a label "01" indicates that EDNS is available. EDNS is the extension mechanism for DNS. This allows the message, which would historically have been limited to 512-bytes, to be extended to be as long as either client or server machine's memory buffer can handle. Setting this flag adds on an additional section to the query/response that further makes DNSSEC possible.

If EDNS is available, the additional section added is the OPT pseudo-section (Vixie, 1999). This section contains three key pieces of information at the present time. The first is the version of EDNS in use (at the time of writing this was "0"). The next is a one bit flag indicating if DNSSEC is okay ("DO"). If the query has the "DO" bit set and the authoritative nameserver has signed its records, it MUST return signed records (Vixie, 1999). If the domain is not signed, then there will be no DNSSEC resource records in the response. The memory buffer size is the final value and this negotiates the size of the DNS datagrams that can be exchanged.

Whichever value is smaller, that from the client or server, becomes the agreed upon size.

The Answer packet sent contains the same header, but it has a different body containing three areas, the Answer, the Authority, and the Additional. Any extended information associated with DNSSEC (a resource record signature, for example) is included in the Additional Section when the extension bit and the "do" (DNSSEC OK) bits are set.

bits 0***4***8***12***16***20***24***28***32	
Answer Name	
Answer Type	Answer Class
TTL	Length of RDATA field
RDATA	
Authority	
Authority Type	Authority Class
TTL	Length of RDATA field
RDATA	
Additional	
Additional Type	Additional Class
TTL	Length of RDATA field
RDATA	

Figure 2.3.: Answer Format for DNS Message

2.2 Attempting to Secure DNS

According to Arends et al. (2005b), "The DNS Security Extensions are a collection of new resource records and protocol modifications that add data origin authentication and data integrity to the DNS." The resource records added included a public encryption key (DNSKEY), a signer (DS), a digital signature for the resource record in question (RRSIG), and an authenticated denial of existence (NSEC) (Arends, Austein, Larson, Massey, & Rose, 2005a). While DNSSEC addresses the integrity issue (and the availability issues that a lack of integrity can lead to), DNSSEC does not, in general, address availability issues or add confidentiality to DNS.

The administrator of a DNS domain is in charge of that domain's zone. When implementing DNSSEC, cryptographic signatures and keys are used to protect the authenticity of the DNS data. The administrator creates two public-private key pairs, one for the zone (Zone Signing Key or ZSK), and one for the key (Key Signing Key or KSK). Each time a new resource record (RR) is added to the zone (such as a new nameserver (NS) record), a cryptographic signature is created for that new record. The zone administrator uses the generated private keys to sign the set of resource records that make up the zone. Then another signature is created, the delegation of signing (DS) record, which is transfered to the parent authority for the zone. This begins the chain of trust up to the root (.) zone. The parent validates the DS record, who in turn has had its DS records validated, all the way to the root. Several top level domains (TLDs) have been signed, but many have not. DNSSEC is only available to those zones whose TLD has been signed. The alternate method of creating a chain of trust was once through DNS look-aside validation (DLV), through which the Internet Systems Consortium (ISC), would provide validation in place of the root signing process. ISC has stopped accepting new zones for validation at this time, and in 2017 will remove all zones from the DLV service.

When a DNSSEC query is made, the records are authenticated using the RRs and the DS records mentioned above. First, an application such as a web browser will request a resource record for example.com. Since an authenticated response is desired, the label for example.com will be initiated with the two bits indicating that EDNS is available (01). There will be an OPT pseudosection included in the query and the "DO" bit will be set. The authoritative nameserver for example.com will respond with an answer packet giving the requested resource record for example.com (likely to be an "A" record if the request originated from a web browser). This resource record will appear in plain text in the answer section. A cryptograpically signed resource record will be included in the additional section. The DNSSEC aware resolver will request the public key for example.com, and a key will be returned. The resolver will do some mathematical checking of the answers. It will hash the "A" record it receives. It will decrypt the signed resource record with the DNSKEY to reveal the signed hash. It will compare the two hash values and will either accept the answer or reject it, depending on whether or not the hashes match.

The process of a resolving DNSSEC aware nameserver querying a DNSSEC signed domain is shown in Figure 1.1 below.

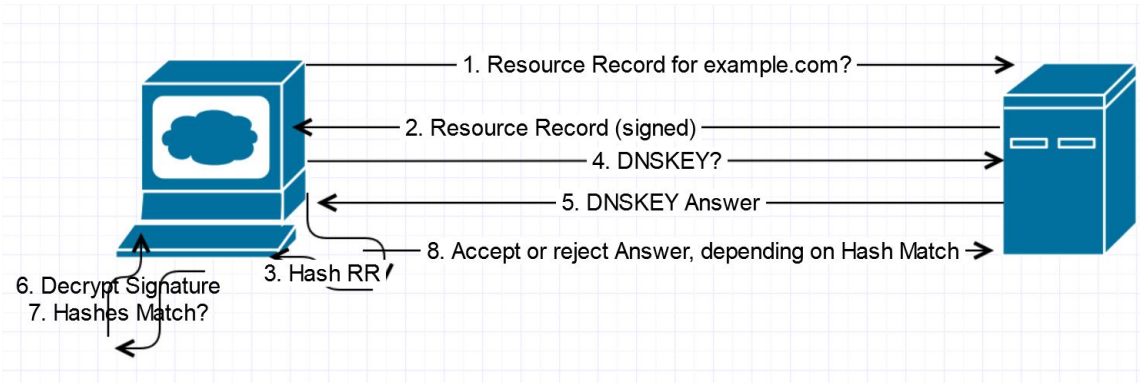

Figure 2.4.: DNSSEC Query Response Validation Process

2.3 The Threats

The usage of DNS by botnet operators and malware writers poses significant challenges to detection and mitigation. Among the consequences of this usage is the utilization of a covert channel that seems particularly effective in achieving the criminal goals of the malware. This is, in part, due to the typical lack of filtration of DNS traffic (Dietrich et al., 2011). If the communication channel of a botnet can be cut off, the lifeline between the botmaster and the botnet is also cut. This underscores the importance of work done to identify communication between nodes in a botnet. It is important to note that computers infected with botnet malware that are unable to communicate with their botmaster are still infected with that malware. Those machines could be a latent threat worthy of consideration in another study.

2.3.1 Botnet Communication and Topology

The earliest botnets utilized IRC as a communication channel between the botmaster and each machine in the net (Norton, 2007). Over time, security professionals have begun to recognize the signs that an IRC channel or user has nefarious intent, and these botnets have been largely mitigated as a result. Additional communication methodologies exist, and malware writers are taking advantage of HTTP, P2P, and DNS to send commands and updates to the bot machines and to exfiltrate user data.

Botnets' topologies have evolved over time, and when considering the potential threat detection and mitigation strategies, one must consider the end goal. Meeting that goal will likely require understanding the structure of the botnet. For some individuals, it is enough to ensure that a single machine is no longer part of the botnet. Doing so may require knowing how that machine is receiving its communications and blocking that process. Should the goal be to cut off communication between the botmaster and the rest of the botnet, certainly

understanding the structure of the botnet will facilitate that. Perhaps most importantly, if the goal is to locate the botmaster, the only way to do that is to understand the path of communications and find the leaked data pointing to his/her/their location.

Many topologies exist in the world of botnets (Ollmann, 2009). The original structure of a botnet was the star topology. In the center, there existed a single Command and Control (C&C) server, and each bot communicated directly with it. There were several advantages, if you were the botmaster, to this approach, including the high speed of control from the central location. No advanced planning was required. Botnets then evolved to utilize multiple C&C servers. These servers would be capable of communicating with one another, and with a sub-section of the bots. Though the multi-server structure allowed for the botnet to continue to operate when one server was located and taken offline, it was still possible to use information leaked in communications between any machine in the botnet to uncover the number and location of the other machines. This included the location and number of C&C machines, leaving the botmaster open to the possibility of being found and held accountable for his/her crimes.

In order to protect against the ability to enumerate the number of bots within the botnet, and to better protect the C&C server from location, a hierarchical structure was created. This succeeds because no bot is able to locate all the other bots. The botmaster is hidden beneath layers of bots. This structure led to the ability to sell portions of the botnet to an interested party while still maintaining control over other branches of the net.

Of the botnet topololgies, perhaps the most difficult for security professionals to shut down is the random topology. This provides a high level of resilience for the botnet due to the fact that when lacking a central C&C, locating the botmaster is near impossible, and any and all other machines can take over the duties of a given machine should it be taken off the network. However, by monitoring the network

communications between the bots from an infected machine, at least the number of bots can be determined.

Since it is quite expensive to set-up and maintain a botnet (Tyagi & Aghila, 2011), resilience is a key quality a botmaster strives for in the design of the botnet. As security professionals found themselves able to locate C&C servers and perhaps send out communications to the bots to kill or uninstall the botnet software, the botmasters looked for means of protecting their communications, their command centers, and thus their financial investment. As is true in information security in general, any tool available to the adversary is one available to the victim - and vice versa.

2.3.2 Building Resilience

Resilience is a key quality a botmaster strives for in the design of the botnet (Gallagher, 2013). Botnet operators and have begun to use a type of service network that creates resilience called Fast-Flux. Fluxing exists in two major groups: IP or Domain-based flux.

2.3.2.1. Domain Flux

Domain flux is the mapping of a single IP address to a set of fully qualified domain names (FQDNs). The botmaster takes advantage of the DNS infrastructure by utilizing one of two types of domain flux. The first is wildcarding subdomains. For example in the domain name *.example.com, the * represents any sub-domain within test.com, such as askjwo.example.com and botnet.example.com. This allows the botmaster to track individual bots on his/her net and to measure the success of his/her malware campaign. The second approach utilizes Domain Generation Algorithms to generate a set of domain names each day that the botmaster will register. One of which will become that day's command and control update location for the malware. When the number of such domains is as large as 50k per day

(Porras, Saidi, & Yegneswaran, 2009), the botmaster has an effective cloak from behind which s/he can hide.

The Conficker worm was an example of botnet malware utilizing a Domain Generation Algorithm (DGA). In its early days, it generated only 250 domains each day (Porras et al., 2009). Though a significant number, it is far fewer than the 50k domains eventually generated daily in later versions (Burton, 2010). Utilizing DNS services to look up each domain name and determine which one would resolve was the worm's calling card to phone home to the C&C server. Once it found the actual domain for the day, it was able to send information to and receive information from the C&C (Burton, 2010). This worm infected millions of machines at one point in its lifetime and it is continuing to send spam from currently infected machines (Selter, 2014). Once the DGA was reverse engineered, global registrars cooperated (ICANN.org, 2010) to block the collection of Conficker domains from registration, thereby blocking the botnet's mechanism for update, command, and control. Conficker was perhaps the first botnet to utilize domains in place of IP addresses, and thus the first to utilize DNS.

Though Conficker is still in the wild infecting machines, it has had to evolve in order to continue to survive due to the collective effort of many registrars, Internet security professionals, and law enforcement around the globe to stop the registration of Conficker domains (ICANN.org, 2010). Once the Conficker Domain Generation Algorithm's functionality was determined, it was possible to predict the set of domains that would be generated on a given day. The set of all such domains were then blocked from registration, theoretically shutting down the botnet's mechanism for update, command, and control. Conficker represented the first

botnet to utilize domains in place of IP addresses, and thus the first to utilize the DNS infrastructure.

2.3.2.2. IP Flux

IP flux is the opposite of domain flux where many IP addresses are mapped to a single fully qualified domain name. By rapidly cycling through multiple IP addresses, the botmaster increases the likelihood of the botnet's survival. IP flux, also referred to as fast flux, comes in two main flavors: single and double flux. The key difference between the two flavors is the use of a "bullet-proof" DNS hosting service/company as the DNS server in single flux, versus the use of a rotating set of bots as proxies in double flux. In a double flux network, the botmaster controls a server that has been referred to as "the mothership" (Salusky & Danford, 2008), which is the C&C hiding under a cloak of infected machine's identities.

Under this extra level of obfuscation in double flux, it is difficult for authorities to locate the mothership to shut down the botnet. But single-flux offers strong protection as well. In the professional experiences of the author detecting and shutting down phishing attacks, it has become clear that the length of time between the detection and the shut down of an attack is largely a function of the cooperativeness and responsiveness of the authority in charge of providing services keeping the fraudulent content online. An attack can be online for years if the authority, such as the one in control of the authoritative name server, is uncooperative.

2.3.3 DNS Covert Channels

A covert channel is a stealthy data-transfer method that avoids typical detection/blockage methods such as firewalls and Intrustion Detection Systems (IDS). In addition to utilizing DNS as a mechanism for hiding the location of the

C&C and increasing the resilience of the botnet, malware writers have also been utilizing DNS as a covert channel (Bromberger, 2011) to hide their existence and whereabouts. DNS signaling and DNS tunneling are two methods utilized.

In DNS signaling, a field within the query can be set to convey a message. For example, a covert channel can be created by using the TTL field. Odd values could signal "yes" and even values would signal "no." As Hoffman, Johnson, Yuan, and Lutz (2012) noted, more than a binary signaling can be accomplished in the TTL field due to the unformatted nature of the field. The digits can represent anything. This would move the use of the TTL field from a signal to a tunnel.

DNS Tunneling represents a more direct threat than DNS Signaling as entire protocols such as HTTP, SFTP, and FTP are encoded. The DNS queries can construct a VPN between the botmaster and bots concealed within the mass of normal DNS traffic (OpenDNS, 2012). In many implementations, the lower level domains in DNS requests made to the authoritative DNS server for a fraudulent domain (or its proxy) contain stolen data such as passwords or account numbers, and the responses can contain commands for updating the botnet software (Rasmussen, 2012). There are several characteristics that can help to identify this type of traffic. As a result of the encoding of data into a subdomain, the domain names being resolved are going to have higher entropy than the typical legitimate DNS query (Zhang, Papadopoulos, & Massey, 2013). The attempt to exfiltrate data in this manner suggests that the appended subdomain will take advantage of the bounds on the length of DNS fields, would be maximized - total character length according to RFC 1035 is 255 bytes, and any single label has a maximum length of 63 bytes. Therefore, labels using the maximum character lengths are suspected of being representative of DNS tunneling traffic (Farnham, 2013). Beyond the use of subdomains, the other resource records are used, such as the TXT record (OpenDNS, 2012). The TXT record is commonly used due to its larger data size, but other fields can contain the covert data as well. The use of the other fields may necessitate a greater number of DNS queries to complete the data transfer,

increasing the likelihood of detection. However, DNS queries are traditionally not monitored and thus, even an excessive number of queries would go unnoticed.

Feederbot, discovered by Dietrich et al. (Dietrich et al., 2011), is an example of malware utilizing DNS signaling as its command and control mechanism. According to the analysis of Dietrich et. al, it sends its upstream messages in the rdata field of the TXT record and it encrypts this data by using the RC4 stream cipher. This use of encryption can be both a method of obfuscation as well as a clue to aid in detection.

Though covert channels over DNS can be used to exfiltrate data, they can also be used to infiltrate data. This threat has not been analyzed in the literature, but the ability of a botnet to send data into a system that can manipulate the information on the system would be damaging to businesses finding themselves unable to trust that their customer data, tax records, accounting information, and/or secret recipes have integrity. Similarly, breaking into an information system and placing an announcement on a key news outlet's social media site could cause stock market panics as Ovide (2013) notes happened a couple years ago.

2.4 Detection and Mitigation of Botnets

When botmasters combine resilient topologies with other advanced malware stealth techniques, such as utilizing covert channels and encryption, the possibility of mitigation and detection seems daunting if not impossible. However, several methods have been utilized to various levels of success in the past.

2.4.1 Deep Packet Inspection

Deep packet inspection represents a highly invasive and privacy destroying method of potential botnet discovery whereby the packet headers and payload are examined. The examination of the payload includes identification of TCP or UDP headers and potentially details the actual data. It is problematic to inspect every

packet sent in an organization in order to potentially locate a botnet because the entity doing the inspection may not be able to be neutral as the content of the traffic is observed (Sloan & Warner, 2013). This could lead to a number of problems that could potentially have a greater impact on a greater number of people than the botnets do. Also problematic for this method is the idea that the content of the packet may be encrypted. Without the key, or a sufficiently weak encryption algorithm allowing for cracking of the key in a reasonable time, this method would be unable to reveal any information beyond that of the headers.

2.4.2 Network Flow Analysis

Network flow records were described by the SANS Institute (Gennuso, 2012) as acting like a telephone bill. They fail to indicate the content of the conversation, but they do communicate who talked with whom. The data contained in a network flow includes the source and destination IP addresses, the router interface used, and the ports for UDP or TCP that were used. There are several protocols for conducting network flows created by proprietary switch and router manufacturers, such as Cisco or Juniper, as well as some open source alternatives. Many botnet location tools have been created utilizing the proprietary protocols, such as Botfinder (Tegeler, Fu, Vigna, & Kruegel, 2012) and Botsniffer (Gu, Zhang, & Lee, 2008). However, there are issues with these programs as a result of the utilization of these proprietary protocols, including the inability to use the tools on products made by other manufacturers. These proprietary protocols rely on a fixed format for defining the flow; in order to model the flexibility of the malware writers, the tool used to locate them must also be flexible.

An open source alternative is called IPFIX. IPFIX was utilized by Wijesinghe, Tupakula, and Varadharajan (2015), et al. in their work to enhance botnet detection. These researchers utilized simulations and virtualizations of bot traffic as well as publicly available botnet datasets to find the features that could be

utilized to recognize infection with a piece of botnet malware. This ultimately relies on patterns to use as signatures for various families of bots. It would not be able to identify a bot on which their model had not been previously trained.

2.4.3 Sinkholing - Spoofing Authoritative DNS Server

Sinkholing has proved to be a powerful tool against known DNS-based botnet threats (Bruneau, 2010). This technique utilizes blacklists (known malware domains) and redirects traffic destined for those domains or for domains that meet criteria set by the administrator of the internal DNS server. Implementing this strategy with rules beyond the known blacklist theoretically allows for the redirection of traffic that could be associated with malware, but simultaneously allows for false positives, which could be detrimental to the end user. In the case of the Polish CERT detailed in (CERT, 2013), the CERT got involved and was able to create a sinkhole that worked to ensure the paths to false positives were protected and thus only actual malware domains were identified. The Polish CERT was also able to use their sinkhole data to identify the registrar operating with the specific intent to allow the registration of domains to be used in illegal activities. They were then able to take it over. This represented an enormous effort on the part of many entities and individuals and could serve as a model for the successful use of a sinkhole in mitigating the botnet malware threat.

It is, however, unlikely that the average network administrator is going to be able to implement this strategy to such a level of success. Sinkholes exist that have been created as commercial enterprises, with their own locations carefully hidden so that malware writers cannot protect against them (Bruneau, 2010). These sinkholes raise their own ethical issues as they are in a position to collect personally identifiable information of the malware victims. This information is part of their

business model, thus making them questionably as unethical as the malware stealing that information in the first place.

2.4.4 DNS Cache Snooping

This technique has strong capabilities of identifying networks utilizing fast flux. Fabian and Terzis (2007) utilized this strategy in order to estimate the lower bound of the size of a fast flux botnet. This technique could reveal some useful information about the structure of a botnet. It can do so by utilizing the partitioning of the Internet by hosts and the DNS servers responsible for them, and by mapping out, within a particular partition, the extent of the connection between a particular domain and a given piece of malware. Again, the threat must be known for this to serve any purpose at all and its utility is limited, at best.

2.4.5 Reverse Engineering

In order to gain an understanding of the functionality of a piece of malware code, without having access to its uncompiled source code, researchers or security personnel can attempt to reverse engineer the code. According to Plohmann et al. from ENISA, via reverse engineering it is possible to classify the malware into families, and occasionally to find signatures for traditional detection (Plohmann, Gerhards-Padilla, & Leder, 2011). The means of installation, communication methods, and means of malware dissemination can be learned.

There are two types of reverse engineering analyses: static and dynamic. If information is learned without executing the binary, then the analysis could be considered to be static. In a carefully controlled environment, one can execute the binary and conduct a dynamic analysis. Changes to the infected system could be measured through this technique, such as attempts to connect to the network, registry files modified, and behaviors such as searching for credentials or keylogging (Plohmann et al., 2011). This requires a large amount of technical skill and is

beyond many researchers. It also requires that one have access to at least the binary of the file, so again, it is only useful in attacks that are known and for which that resource is available.

2.4.6 Honeynets

A honeynet is a very comprehensive strategy for protecting a network from intrusion, including against social engineering attempts (Munro, 2015). Social engineering can be a means through which botnet malware is spread. Honeypots are single machines set up to be purposefully exploitable and attractive to attackers. A honeynet is a network of at least two such machines. In order for this setup to be comprehensive, the machines must appear to be used by a real person such that the attacker will believe there is possibly valuable personal or corporate information to obtain by breaking in to that machine or network. The creation of a honeynet can allow for the collection of malware samples (Martin, 2001) and thus, any of the reverse engineering benefits can be gained from that sample.

2.4.7 Statistical Packet Analysis

Using statistical analysis of the text contained within the fields of DNS queries, multiple studies use an entropy measure of the encrypted text in the covert channel to identify anomalous traffic (Born & Gustafson, 2010; Dietrich et al., 2011). Machine learning algorithms are used strongly in these methods of detection of anomalous traffic. The basic idea is that the distribution of letters would be different in encrypted text than in plaintext, and thus by determining that distribution statistically, one could determine, without direct, invasive deep packet

inspection, the content of the packet and whether it likely came from a human or the bot.

2.4.8 Topological Data Utilization

Dagon et al. (Dagon, Zou, & Lee, 2006) investigated the importance of time zones in malware operations. They determined after studying dozens of botnets over a period of six months that these malware samples demonstrated clear day-night (diurnal) behavior. As recognized by Porras et al. (2009) regarding Conficker, Dagon et al. noted that many botnets display a regional preference/avoidance mechanism for execution. In the case of Conficker, if the malware found that it was located in the Ukraine (by use of the Ukrainian keyboard), the code would exit and not execute. The day-night patterns could help identify botnet malware's presence as well as the location of the infected machines geographically. As with many of the other techniques, this is not effective against zero-day attacks.

One clue that arises from bots that query regularly but receive active instruction from their C&C only during particular time periods is that of the presence of the loopback and RFC 1918 addresses in a response to an external domain question. This answer could be utilized by the botnet malware in a number of ways. It would serve as an acknowledgement that the C&C was still available, but did not have instructions for the bot at the time. It would also be a way of the bot exfiltrating data through the question query without giving an IP address location of the authoritative name server for the fraudulent domain that might easily appear in someone's logs. In either case, this exposes an attribute of covert channel traffic data: the presence of the loopback or RFC 1918 addresses.

2.4.9 Passive DNS

Florian Weimer introduced an architecture for the passive collection of DNS records in 2005 (Weimer, 2005). It was proposed that though Mockapetris (1987)

suggested that the entire DNS database could never be known, a subset of the database could be known. DNS resolvers were recruited to participate and the records of DNS queries they handled could be replicated and stored. The passive DNS records collected are accessible to security researchers and network administrators. A reconstruction process takes place in order to sort the received DNS queries into three separate categories: `UDP_QUERY_RESPONSE`, `UDP_UNANSWERED_QUERY`, and `UDP_UNSOLICITED_RESPONSE` (Edmonds, 2012). Edmonds (2012) reports that this filtering helps make the passive database robust against a spoofing attack. Additional filtering of the data is done, including discarding all but the `UDP_QUERY_RESPONSE` records, any record with the truncated (TC) bit set (which means that the query would have been resent over TCP). The processing of the data collected by the passive DNS server participants is extensive, and unfortunately for the purpose of detecting botnet communication, many lookups that resemble tunneled lookups are discarded in the static filtering process (Edmonds, 2012), however these discarded queries are sent to a discard channel where they may be able to be identified. The passive DNS database also makes raw, real–time data available and thus the duplicated traffic and discarded queries would be available from that.

2.4.10 Active DNS Querying

There are a number of large organizations such as SURF (van Rijswijk-Deij et al., 2014) and CAIDA working to make active queries of the IPv4 and IPv6 routed space. These queries are used in order to map domains to IP addresses. This is done in order to determine the bailiwicks of various domains, to determine the size of the Internet, among other goals. CAIDA mapped queries to the entire IPv4 routed space. However due to the highly structured and purposeful nature of the queries that were created, the responses received do not help identify botnet traffic - the queries were legitimately created.

2.5 Detecting and Mitigating Botnet DNS Covert Channels

Several individuals have written software to create DNS tunnels for varying reasons and purposes. OzymanDNS, created by Dan Kaminsky (2004), and DNSCAT2 (Bowes, 2015) create SSH tunnels over DNS. Additional tunneling tools may attempt to encapsulate other protocols such as TCP, HTTP, or IP over DNS. These tunnels essentially create a VPN allowing for the transfer of non-DNS data over the DNS protocol and utilizing DNS servers. Further examples of tunneling applications include dns2tcp, Iodine, and Heyoka. Each functions slightly differently, either implementing encryption or not, utilizing different types of DNS queries to create the tunnel. Of the freely available tunnel tools, DNSCAT2 has been specifically created to act as command and control. It is meant to be used as an offensive security tool.

2.6 Discussion

Though Born and Gustafson (2010); Couture (2010); Dietrich et al. (2011) and Farnham (2013) have investigated identifying DNS tunnels, none has reported on the creation of a tunnel over DNS when the security extensions to DNS are used. As DNNSEC is implemented at an accelerating pace, if tunnels are created it is key that they are able to be identified such that perhaps the damage they cause could be mitigated. This represents a significant gap in the research and this study sought to fill that gap by determining what theoretical covert channels exist and by demonstrating a proof of concept DNSSEC enabled covert channel.

2.7 Summary

Despite the fact that botnets and their command and control have been investigated for many years, a gap in the literature has been identified. The communication methods of the bots have changed over time as infrastructure and technology have changed. As DNSSEC was first introduced in 1999 by

D. E. Eastlake et al. (1999) and in its present form by Arends et al. (2005a, 2005b), but has recently begun to gain momentum in adoption as shown on the DNSSECSTAT website Lamb (2016). With this change in rate of adoption, it is essential to ensure that network administrators continue to have protection against the exfiltration and infiltration of data by botnets or other malware that may decide to use DNSSEC.

CHAPTER 3. FRAMEWORK AND METHODOLOGY

3.1 Study Design

This final study design diverges from the methods and design put forth at the time of the thesis proposal. The originally proposed research design sought to investigate the question of whether or not methods in use by network administrators to identify covert channels/tunneling over DNS would also identify tunnels over DNSSEC. This entailed the use of Intrusion Detection Systems' (e.g. Snort, Security Onion, or Splunk) rules for identifying DNS tunnel traffic and the application of those rules to real world DNS traffic. The real world DNS traffic needed to have the characteristics of containing both DNS and DNSSEC traffic, it needed to have the potential to have both malicious and benign traffic, and it needed to be able to represent global usage of DNS. This data was not available early enough for use in writing this document. Therefore, the study was modified to focus on determining what features a covert channel in DNSSEC would have and to determine if DNSSEC specific covert channels could exist.

As the modified study sought to determine the possible means of creating covert channels in DNS and DNSSEC, the differences between the two versions of the DNS protocol were examined.

DNS tunnels are commonly used by individuals for non-nefarious purposes, though these uses nearly universally are implemented to circumvent some security mechanism (and thus still match the definition for a "covert" channel). The commonplace usage of tunnels, however, means that a number of freely available DNS tunnel software programs exist. These are open source, which makes them available for "looking under the hood" to determine how they accomplish their

tunneling mechanisms. The study looked at the code implementation of two different DNS tunneling programs, DNSCAT2 and Iodine. DNSCAT2 was released in December of 2015 with the stated purpose of being for Command & Control (Bowes, 2015). The code is well documented. Iodine was created for the stated purpose of creating an IPv4 tunnel to allow Internet browsing where traffic may be firewalled, but DNS traffic is allowed (Ekman, Anderson, & Bezemer, 2014).

The protocol analysis was cross referenced with the functionality of DNSCAT2 and Iodine with the DNS and DNSSEC protocol functions, specifically as implemented by BIND. These two tunneling tools are both open source whose code bases have recently been updated and that implement advanced techniques for creating tunnels, such as encryption in DNSCAT2 (Bowes, 2015) and such as VPN creation in Iodine (Ekman et al., 2014). The functionality of PSUDP (Born, 2010) was also considered, however there is no code base available openly.

A domain name was secured from a commercial registrar. Two custom name servers were configured using Bind 9 and Ubuntu. One, a physical server, served as the master nameserver and one, a virtual machine, served as the slave. Both were configured to be authoritative only and were not configured to handle external recursive queries. Upon configuring the zone for this domain, the appropriate cryptographic keys were generated and the resource record and zones were signed. A delgation of signing (DS) record was created and entered into the registrar's public interface. A virtual "bot" machine was created that also runs Ubuntu 14.04. This machine served as the client in the Iodine and DNSCAT2 tunnel examinations. Further, it served as the client in the prototype testing. The virtual machines were hosted on a Windows 10 server. Figure 3.1 below illustrates the network architecture of the constructed lab environment.

"Dig," an open source tool that stands for "Domain Information Groper" was used to test DNSSEC configurations. By carrying out domain name queries against the custom domain, the DNSSEC results could be examined.

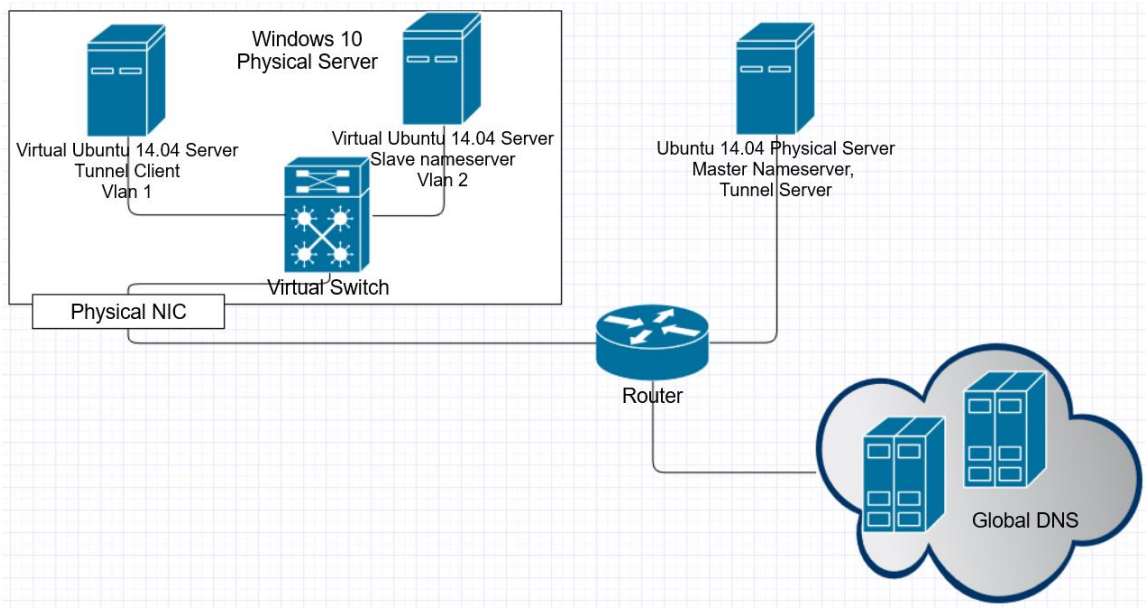

Figure 3.1.: Research Lab Architecture

Each DNS query / response can either be a message for the creation / use of a covert channel or a it can be a legitimate DNS query. In efforts to detect covert communications, the query in whole or in part, alone or as part of a larger chunk of network traffic, would need to be examined.

When certain changes are made to an authoritative name server, these changes are logged in the systems administrator log ("\var\log\syslog"). Observations were made of the entries in the master nameserver syslog as well as the slave nameserver syslog, and were correlated with observations of the results of "dig" queries to determine whether or not the change registered outside of the nameservers.

3.1.1 Hypothesis

The study sought to determine if the possibility existed for a unique covert channel in DNSSEC that did not exist before the security extensions were added.

The following hypothesis was examined: Unique covert channels are present in DNSSEC.

3.1.2 Population

The population for this study was the set of all DNS queries and responses (typically in pairs, though not always) that would be handled on a given DNS client or server.

3.1.3 Sample

The nameservers for the specially registered domain were configured to handle only queries for the domain for which they are authoritative. That is to say, they did not act to forward queries on to other nameservers for which they do not have answers. As the domain was not publicly registered with any search engines and was not hosting any content, it was highly unlikely that any queries would be coming in to the nameservers by individuals other than the researcher (however it was not impossible). The sample was, therefore, limited to the queries specifically made by the researcher to the domain and the responses received.

Due to the nature of the covert channel investigated, there are ten digits available and eight of them are needed to encode a single character. During the trial, one character and a fraction of the next character was encoded into the available digits. As the chosen string was eight (8)characters long, this translated to a 64 digit long encoding, requiring seven (7) queries to transfer.

3.1.4 Variables

A string of text was created to use for the data transfer process. This text, specifically the string “P@$$w0rd” was chosen to represent sensitive, unauthorized data that could be transmitted by a covert channel. This string was chosen due to

the fact that the word “password” is one of the most common passwords in use. Many password rule sets require a capital letter, a symbol, a number, and lowercase letters. Therefore, the plain text “password” was modified to fit those rules using symbols and numbers that continue to represent the letters they replace.

The string can be sent in plain text, can be encrypted, or encoded into hex, base32/64, or binary. The string representation within a tunnel is contingent on the formatting of the packet field and on possible means of obfuscating the traffic beyond the covert channel. For example, the given string converted to binary is the value “01010000010000000010010000100100011101110011000001110010011001 00”. Depending on the channel created, a different representation of the message would be appropriate and/or necessary.

If the string is too long to be encoded into a single DNS query/response packet, the string would need to be split into chunks of appropriate size to each to be sent through the covert channel. The appropriate number of messages would then need to be sent based on the total length of the text.

Further, there is a length of the field into which the text would be encoded. This factors in to the number of messages that would need to be sent in order to encode the entire stolen string.

3.1.5 Threshold

A covert channel is considered to be covert if it matches the definition set forth by Tsai et al. (1990) of a covert channel. It is as follows:

> “given a nondiscretionary security model M and its interpretation $I(M)$ in an operating system, any potential communication between two subjects $I(S_i)$ and $I(S_j)$ of $I(M)$ is covert if and only if any communication between the corresponding subjects $I(S_i)$ and $I(S_j)$ of the model M is illegal in M” (p.1).

If a communications channel is able to transfer information between two unauthorized parties, it would be considered to have construct validity with respect to this definition of a covert channel.

Due to the specific configuration of the test lab, the external validity (generalizability) is presently limited for the creation of the given covert channel. In order to ensure greater generalizability, future work will need to be conducted by creating different test labs with domains that are signed by their registrars or that use other DNS implementations besides BIND.

There is significant variation expected in the amount of time it takes for a change in a DNS record to propagate through the system. Therefore, the timing of these results would be subjected to this random variation and would be a factor in the reliability of the results.

3.2 Summary

This chapter provided the framework and methodology to be used in the research study.

CHAPTER 4. RESULTS

4.1 Description of Data

Data was collected on existing covert channel software, on the DNS protocol, on the DNSSEC extensions to the DNS protocol, and on the results of an attempt to build a prototype covert channel using DNSSEC. The examined hypothesis predicted that DNSSEC specific covert channels exist.

4.1.1 Existing Covert Channels

There are a number of mechanisms that are currently in use by covert channels to encode data in a DNS message. DNSCAT2 has two mechanisms for creating a connection between the client and the server. The first is a direct connection without the use of name resolution. The user inputs an IP address and a port, and traffic is sent between the two machines connected at that IP:Port combination. This traffic is prepended by a tag with the text “dnscat". The DNSCAT traffic, in the direct connection scenario, using only the TXT record. The query would make a request with Query ID (QID) 0x5de4 for the TXT record for “dnscat.2548700ebb14859f44456d6d6d6e6420". The response would be an answer with QID of 0x5de4 and record type TXT with no content. The data in the direct connection case is in the position of the Top Level Domain.

However, when the connection was created requiring a domain lookup, the traffic shown in Figure 4.1 below was captured. The records used include CNAME, MX, and TXT. Data was encoded bi-directionally. This data is all encrypted using the Salsa 20 suite of cryptographic tools. The encrypted data is encoded in the

Lower Level Domain location (LLD) in these queries, and the domain (frankfrank.org) is in its traditional location in the queries and responses.

Figure 4.1.: DNSCAT Traffic Capture

When using the Iodine software, the DNS packets use the NULL question type. The data is base64 encoded into the Lower Level Domain (LLD) and appended to frankfrank.org. The packet sizes captured ranged from 113 bytes to 937 bytes.

In both the case of Iodine and of DNSCAT2, the data is encoded into the domain name in a non-human readable format. The implementations of the tunnels differ in the DNS question and answer types they choose to use to frame the data.

4.1.2 Covert Channel Prototype

A previously undocumented covert channel was created using the "serial" field of the SOA record for the DNSSEC signed domain. A string was encoded into binary and its binary digits were placed into the serial field of the zone file. The zone was resigned, the bind service was reloaded, and then a query was made to the domain by the client for the SOA record. The SOA record displayed the encoded message on the client machine. The figure below diagrams this process.

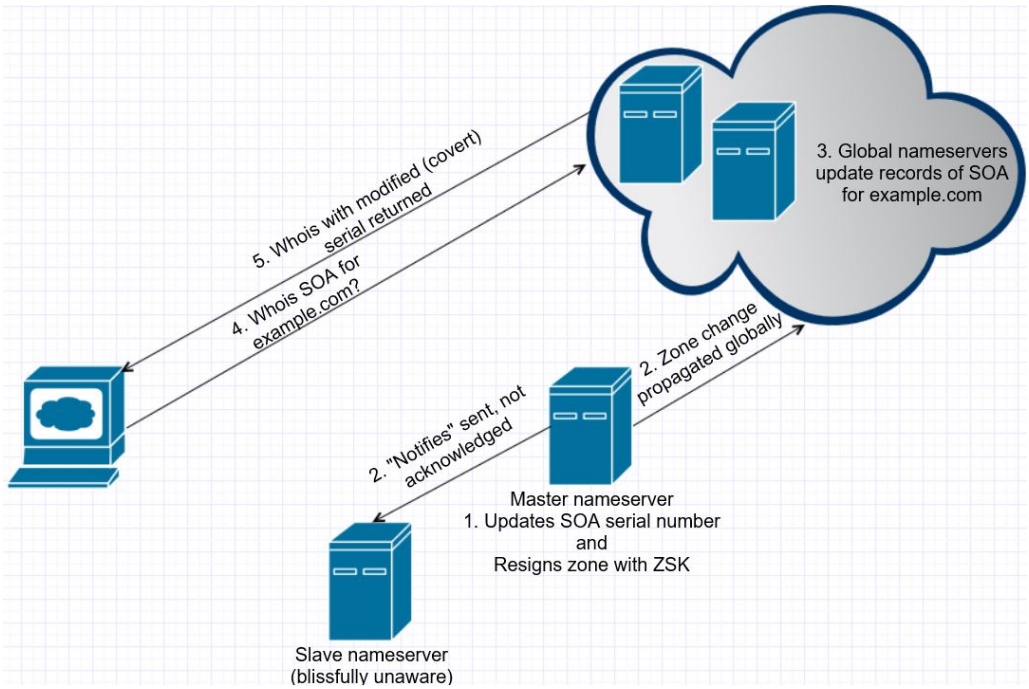

Figure 4.2.: SOA Serial Field Covert Channel

CHAPTER 5. CONCLUSIONS, DISCUSSION & RECOMMENDATIONS

5.1 Research Question

What features of DNSSEC could be exploited to create DNSSEC specific covert channels? What characteristics would such a covert channel have? The study investigated the hypothesis that DNSSEC specific covert channels exist.

5.1.1 DNS & DNSSEC Protocol

Past covert channels provide clues to ascertain what new covert channels may look like. The features of the protocol that are or could be exploited were considered. In so doing, it is important to note that both DNS and DNSSEC are the same DNS protocol, they are not two separate protocols. DNSSEC exists due to the addition of new record types and an expanded packet size in the original DNS specification. All the fields in a DNS packet exist in a DNSSEC packet. The Z field in a DNS packet contained 3 bits all set to zero. With the addition of DNSSEC, the second and third bit can each be set to 1, depending on the query and response to indicate Authenticated Data (AD) and Checking Disabled (CD), respectively. When the DNS question label has its first two bits set to 01, the EDNS (extensions to DNS) allow for the addition of the OPT pseudo-record, which includes a bit flag for “DNSSEC OK" (DO).

EDNS can be used without DNSSEC. In those cases, it allows for the creation of larger packets without the need for truncation or switching over into TCP from UDP. Already existing in the DNS protocol is the possibility of tunneling data through the domain label, typically through the lower level domains. This happens

in the answer field of A and AAAA records, CNAME, MX, TXT, and NULL record types, as demonstrated by two implementations of tunnels. There are many Resource Record types, and several have been used by other tunnel implementations (such as the now depricated KEY record by DNS2TCP or NS by the older version of DNSCAT (Farnham, 2013)). The TTL field has been and can be used to create a covert channel due to its unformatted nature Hoffman et al. (2012).

The RFCs associated with DNS and DNSSEC set forth requirements that MUST and SHOULD be met in any DNS implementation. One such requirement is that the RCODE (return code)field is ignored in queries as it is a field that the server expects to define. A specially configured (i.e. non-conforming) tunnel server could check that field for data and that would serve as a signal (or a small tunnel as the RCODE field is 4 bits). This is not specific to DNSSEC, but it is worth noting that this signal could exist and also could be observed relatively easily (likely diminishing its usefulness as a covert signal). As DNS tunnel software typically captures all incoming DNS traffic and forwards the traffic not relevant to the tunnel on to the traditional DNS software, the tunnel server can be configured to be nonconforming to specifications (and thus willing to accept packets that RFCs state “MUST" or “SHOULD" be thrown away) while still having a conforming nameserver running on a machine handling all non-tunnel queries.

Born (2010) proposed a very different type of tunnel driver that does not generate its own traffic, but uses slack space in legitimate traffic to hide its own content. Pointers are used within the DNS packet to allow for compression. Whereas a traditional DNS name label starts with “00", the extended label begins with “01", a label beginning with “11" indicates a pointer. The pointer will be followed by an offset indicating where the DNS message parsing software should read next. Traditionally, this is meant to point back to the first instance of the domain name label in the packet (and thus the pointer would point to a part of the packet before the pointer), however there is nothing in the RFC describing compression Mockapetris:1987 that requires this. Therefore, Born argued, the

pointer could point to the end of the packet where exfiltrated data is appended and then retrieved by tunnel software. Extending this into the realm of DNSSEC, each resource record contains a reference to the label. Each of these labels can be replaced by a pointer. As a result of the data compression, slack space would be created within the packet where covert data could be hidden. As a DNSSEC packet can be as large as the resolvers' memory buffers on either end of the message exchange can handle, the amount of slack created by compression would have increased size potential in a DNSSEC packet than in DNS alone. This mechanism is not unique to DNSSEC, it would just be amplified by it. Such a tunnel would be devastating to DNSSEC whose adoption has already been slow.

DNSSEC has been shown to protect against spoofed domains and cache poisoning (P. Silva, 2009). It remains susceptible to DNS amplification attacks, as shown by van Rijswijk-Deij et al. (2014). It provides a measure of protection against covert channel usage where a question is asked in the A or AAAA record for somecrazyencodedtext.example.com, due to the fact that the response would be an authenticated non-existent domain (NXdomain) record. This does not stop "somecrazyencodedtext" from being received by the rogue authoritative nameserver, but, with the DS records and chain of trust fully in place, it does make it difficult to return an encoded answer.

None of these methods of creating tunnels thus far discussed are DNSSEC specific, though it seems that DNSSEC does little to mitigate data exfiltration via tunnel.

5.2 Conclusions

Do DNSSEC specific tunnels exist? This study has determined the existence of at one such tunnel, though it does not rule out the possibility of others.

When a domain is DNSSEC enabled and when a change is made to its zone's records, such as the addition of a new name server or a change in IP address, the

zone must be resigned. In order for the zone change to be communicated from the master to the slave nameserver, a field in the Start of Authority (SOA) resource record called the “serial" must be incremented. The serial field is a ten digit number. When configuring the master nameserver for frankfrank.org and observing system logs, it became clear that a failure to increment the serial field would, as expected, fail to trigger a zone transfer to the slave nameserver, even if the change had been signed. However, a dig query of the SOA for the domain returned the updated zone values. This observation suggested that the serial field could be used as a covert channel, and could be used in a hijacked master nameserver without alerting the system administrator with zone transfers.

The master nameserver will send a “notifies", but the slave nameserver will not acknowledge it. Traditionally, the serial number is given a value to indicate the date it was generated and the version change for that date, though there is no rule requiring this format. With that form, for the first change made on July 16, 2016, the serial number would take on the value “2016071601." In order to ensure that the value is lower than this value, binary encoding of a stolen text could be used.

Once the data that one wants to send is obtained, it is then converted to binary. In the initial trials, ten bits at a time were transmitted, with the final message containing four bits was sent as only four bits. Three commands were executed. 1. Modify the serial number in the zone file. 2. Resign the zone. 3. Reload the BIND service. In a time from of between 90 seconds and seven (7) minutes, a whois lookup reflected the changed serial number. No slave nameservers were alerted to the change. This was repeated a sufficient number of times to encode a theoretical sensitive, stolen data string (“P@$$w0rd”), which is 64 bits long in binary. This channel encodes essentially one character per zone change. The results of the whois queries are included as Appendix A. Any serial numbers with leading zeros had those zeros truncated.

Though the serial field exists in the SOA record of the base DNS query, the mechanism by which the global database is forced to recognize the update to a

serial number that may or may not be an increment of the previous one is through the zone signing action. The covert channel demonstrated in Appendix A would not be possible without the DNSSEC zone signing process. Therefore the author rejects the null hypothesis that there are no unique covert channel opportunities available in the security extensions of DNS.

This prototype/proof of concept demonstrates that data transfer using the SOA serial field is possible. However, there are a number of issues that would need to be addressed long term if this covert channel were going to be built into a tunnel application. The first, is that the prototype does not implement a method for decoding on the client side. As text characters typically have 8-bit binary values and the data field is 10 bits in length, total data lengths other than multiples of 80 bits will need to account for padding. This is especially true when leading zeros are removed from full 10 bit transfers, making it difficult for the client to discern whether a given packet is the final in the transmission or one requiring the addition of padding with initial zeros.

A solution to this would be to encode the first bit of the serial to "1". This serves the purpose of ensuring that any leading zeros will not be truncated and can serve as a signal that a covert message is incoming. The final bit in the serial would be set to either the number 1 or 0, with this signaling that either another serial number is incoming or that this is the final message in the stream, respectively. With these two bits sandwiching the encoded character, each 8 bit value would be encoded in a single packet and the recombination of the message on the client side would be facilitated. The figure below encodes a simple three letter word ("and") to demonstrate how this padding would work

The transfer is slow. Stolen data would need to be chosen carefully, and the model of attacker would need to be a patient one. The data uploading was, for purposes of the proof of concept, carried out by hand. This would eventually be implemented programatically. However, because DNS updates can take varying amounts of time to propagate, an acknowledgment would be needed from the client

Covert Message Encoding			
Position	Initialize	Message	Terminate
First message (“a”)	1	01100001	1
Middle message (“n”)	1	01101110	1
Last message (“d”)	1	01100100	0

Figure 5.1.: Encoding for Serial Covert Channel of the Word “and”

before the next zone update could be made to be sure that the change had reached its intended audience. As it stood, the entire password transfer took approximately 22 minutes, with the longest delay between transfers having been caused by human error between transfer 1 and 2. After a rhythm was obtained, transfers occurred fairly regularly at about 2 minute intervals (at :11, :13, :15, :17, & :19 minutes after the hour).

The SOA serial channel represents a unidirectional transfer, such that the data moves from server to client, and not in the more traditional direction. However, as discussed in earlier sections, data infiltration and modification can seed distrust in an information system, and possibly worse. As Ovide (2013) reported, misinformation placed on a social media feed caused a stock market panic. More carefully coordinated misinformation campaigns could decide democratic elections or lead people to riot. Therefore, despite its clear shortcomings, such a covert channel could still be powerful.

Attempts were made to replicate this result with changes to the serial field in the frankfrank.org zone file without initiating the resigning process, and the propagation did not occur. The author was unable to obtain these results using an unsigned domain and serial record changes alone. This demonstrates that the mechanism making this covert channel possible is the zone signing process. In a signed domain, any change made to the zone without a resigning would be invalid. Therefore, this implementation is absolutely appropriate to maintaining the integrity

of the zone records. However, this suggests that other implementation details could open the way for covert channels not necessarily based in the new Resource Records associated with DNSSEC but rather in other implementation methods. The SOA record contains several values which indicate times of creation, expiration, refresh, etc. of the SOA record. It is unknown, but highly likely, that the use of these fields to encode data could lead to a larger covert channel. This is left for a future study.

It is unclear whether or not these results could have been obtained had the chain of trust been established for frankfrank.org. Attempts to test this without access to a validated, signed domain proved to not be possible. However, specific "dig" queries to frankfrank.org with the "+dnssec" flag from the lab's bot machine returned all the signed resource records despite the "AD" (Authenticated Data) bit not being set. Since the data can be obtained in this way, these results indicate that a client requesting this information will receive the covert information without any of the typical characteristics that IDS's and SIEM's look for, such as increased entropy in domain names or very long label lengths. This is not presently a covert channel mechanism that IDS's and SIEM's are configured to identify and block. A non-conforming tunnel program running on a client machine could be programmed to recognize this result as a possible covert channel entry query.

5.3 Recommendations

Frequent updates to a zone are not abnormal, especially in large zones. It has been recommended that administrators resign their zones once per day. With this in mind, creating an IDS or SIEM signature that looks for rapid changes in serial numbers may or may not be an effective approach to dealing with the demonstrated covert channel. Since the bandwidth associated with this channel is very low, and the latency is also very high, any attacker willing to use this method would be a patient one. A patient attacker could easily modulate the zone changes to be fewer than whatever threshold is decided does not block legitimate secure

zones. This model of patient attacker needs to be kept in mind as covert channels are considered in general. Often, low bandwidth channels are dismissed offhand because data moves slowly through them, but data does move, nevertheless.

In the future, covert channels using DNS will, as the scenario above suggests, evolve to avoid current detection mechanisms. Packets that create channels will emerge that do not resemble those currently generated by DNSCAT2 and Iodine, with long, random subdomains whose character frequencies and entropy values differentiate them from normal, legitimate DNS traffic. As discussed, non-conforming DNS servers and resolvers can be configured to break RFC guidelines of what "MUST" and "SHOULD" be done with certain types of DNS packet. Intrusion Detection Systems presently have methods of identifying such nonconforming traffic and blocking it before it reaches the nameserver. Future tunnel traffic will conform to the RFC guidelines, and thus is unlikely to be stopped by these methods. The cat and mouse game will eventually select for covert channel traffic that is truly covert in that it will be impossible to separate signal from noise. Therefore, as the security community pushes for stronger safeguards of information, DNSSEC adoption is a good place to start, however it does not go far enough to protect against all the known attacks on DNS. With the discovery of the "serial" covert channel, it also provides new vectors of attack. Therefore, additional security measures will be needed to protect DNS long-term.

5.4 Significance

This study provides a methodology for the creation of a covert tunnel and the examination of the results. This methodology could be used to test the covert channel's effectiveness on different DNS implementations, such as that by Windows.

The study is also significant in that it documents a previously undocumented covert channel over DNS, and specifically over DNSSEC. This can immediately be put to use by systems administrators who monitor their DNS traffic. Systems

administrators can monitor the number of SOA records being requested and received from a given domain, and can potentially catch the covert transfer of data through the repeated use of this DNS record.

LIST OF REFERENCES

LIST OF REFERENCES

Arends, R., Austein, R., Larson, M., Massey, D., & Rose, S. (2005a). *Dns security introduction and requirements* (Tech. Rep.).

Arends, R., Austein, R., Larson, M., Massey, D., & Rose, S. (2005b). *Protocol modifications for the dns security extensions* (Tech. Rep.).

Baranowski, S. (2003). *How secure are the root dns servers?* Retrieved from https://www.sans.org/reading-room/whitepapers/dns/secure-root-dns-servers-991

Born, K. (2010). Psudp: A passive approach to network-wide covert communication. *Black Hat USA*.

Born, K., & Gustafson, D. (2010). Detecting dns tunnels using character frequency analysis. *arXiv preprint arXiv:1004.4358*.

Bowes, R. (2015). *Dnscat2.* Retrieved from https://github.com/iagox86/dnscat2

Bromberger, S. (2011). Dns as a covert channel within protected networks. *National Electronic Sector Cyber Security Organization (NESCO)(Jan., 2011)*.

Bruneau, G. (2010). *Dns sinkhole.* Retrieved from http://www.sans.org/readingroom/whitepapers/dns/dns-sinkhole-33523

Burton, K. (2010). *The conficker worm.* Retrieved from https://www.sans.org/security-resources/malwarefaq/conficker-worm.php

CERT. (2013). *Report.* Retrieved from http://www.cert.pl/PDF/Report_CP_2013.pdf

Comey, J. (2015). *Statement before the senate committee on homeland security and governmental affairs.*

Couture, E. (2010). Covert channels. *SANS Institute*.

Dagon, D., Zou, C. C., & Lee, W. (2006). Modeling botnet propagation using time zones. In *Ndss* (Vol. 6, pp. 2–13).

Dietrich, C. J., Rossow, C., Freiling, F. C., Bos, H., van Steen, M., & Pohlmann, N. (2011). On botnets that use dns for command and control. In *2011 seventh european conference on computer network defense* (pp. 9–16).

Eastlake, D., & Kaufman, C. (1997). Rfc 2065: Domain name system security extensions. *Obsoleted by RFC2535*.

Eastlake, D. E., et al. (1999). Domain name system security extensions.

Edmonds, R. (2012). Isc passive dns architecture. *Internet Systems Consortium, Inc., Tech. Rep*.

Ekman, E., Anderson, B., & Bezemer, A. (2014). *Iodine readme.* Retrieved from http://code.kryo.se/iodine/README.html

Fabian, M., & Terzis, M. A. (2007). My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging. In *Proceedings of the 1st usenix workshop on hot topics in understanding botnets, cambridge, usa.*

Farnham, G. (2013). *Detecting dns tunneling.* Retrieved from https://www.sans.org/reading-room/whitepapers/dns/detecting-dns-tunneling-34152

Fulton, S. (2015). *Top 10 dns attacks likely to infiltrate your network.* Retrieved from http://www.networkworld.com/article/2886283/security0/top-10-dns-attacks-likely-to-infiltrate-your-network.html

Gallagher, S. (2013). *A beginner's guide to building botnets - with little assembly required.* Retrieved from http://arstechnica.com/security/2013/04/a-beginners-guideto-building-botnets-with-little-assembly-required/

Gennuso. (2012). *Shedding light on security incidents using network flows.* Retrieved from http://www.sans.org/readingroom/whitepapers/networkdevs/shedding-light-securityincidents-network-flows-33935

Gu, G., Zhang, J., & Lee, W. (2008). Botsniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th annual network and distributed system security symposium, ndss.*

Hoffman, C., Johnson, D., Yuan, B., & Lutz, P. (2012). A covert channel in ttl field of dns packets. In *Proceedings of the international conference on security and management.*

Huq, N. (2015). *Follow the data: Analyzing breaches by industry.* Retrieved from https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/follow-the-data

ICANN.org. (2010). *Conficker summary and review.* Retrieved from http://icann.org/en/security/conficker-summary-review-07may10-en.pdf

Kaminsky, D. (2004). Black ops of dns. *Black Hat Briefings*.

Lamb, R. (2016). *Dnssec deployment report.* Retrieved from https://rick.eng.br/dnssecstat/

Martin, W. (2001). *Honey pots and honey nets: Security through deception.* Retrieved from http://www.sans.org/readingroom/whitepapers/attacking/honey-pots-honey-nets-securitydecption-41

Mockapetris. (1983). *Rfc 1034: Domain names: Concepts and facilities.* Retrieved from https://www.ietf.org/rfc/rfc1034.txt

Mockapetris. (1987). *Rfc 1035: Domain names: Implementation and specification.* Retrieved from https://www.ietf.org/rfc/rfc1035.txt

Mozur, P. (2014). *China websites hit with disruptions.* Retrieved from blogs.wsj.com/digits/2014/01/21/ chinas-sina-baidu-and-other-bit-websites-are-hit-with-disruptions/

Munro, K. (2015). *How to build intelligence for your business by creating a honeypot.* Retrieved from http://www.techradar.com/us/news/world-of-tech/how-tobuild-threat -intelligence-for-your-business-by-creating-ahoneynet-1283368

Norton. (2007). *Bots and botnets - a growing threat.* Retrieved from http://us.norton.com/botnet/

Ollmann, G. (2009). Botnet communication topologies. *Damballa*.

OpenDNS. (2012). *The role of dns in botnet command & control.* Retrieved from http://www.securityweek.com/ do-you-know-what-your-dns-resolver-doing-right-now

Ovide, S. (2013). *False ap twitter message sparks stock-market selloff.* Retrieved from http://www.wsj.com/articles/ SB10001424127887323735604578440971574897016

Plohmann, D., Gerhards-Padilla, E., & Leder, F. (2011). Botnets: Detection, measurement, disinfection & defence. *The European Network and Information Security Agency (ENISA)*.

Porras, P., Saidi, H., & Yegneswaran, V. (2009). An analysis of conficker's logic and rendezvous points. *Computer Science Laboratory, SRI International, Tech. Rep*.

Rasmussen, R. (2012). *Do you know what your dns resolver is doing right now?* Retrieved from http://www.securityweek.com/ do-you-know-what-your-dns-resolver-doing-right-now

Salusky, W., & Danford, R. (2008). Know your enemy: Fast-flux service networks. an ever changing enemy. *The Honeynet Project*.

Selter, L. (2014). *Conficker: Still spamming after all these years.* Retrieved from http://zdnet.com/article/conficker-still-spamming-after-all-these-years/

Silva, P. (2009). Dnssec: The antidote to dns cache poisoning and other dns attacks. *A F5 Networks, Inc. Technical Brief*.

Silva, S. S., Silva, R. M., Pinto, R. C., & Salles, R. M. (2013). Botnets: A survey. *Computer Networks*, *57*(2), 378–403.

Sloan, R. H., & Warner, R. (2013). *Unauthorized access: The crisis in online privacy and security.* CRC Press.

Tegeler, F., Fu, X., Vigna, G., & Kruegel, C. (2012). Botfinder: finding bots in network traffic without deep packet inspection. In *Proceedings of the 8th international conference on emerging networking experiments and technologies* (pp. 349–360).

Thakar, N. (2013). *Botnets remain a leading threat.* Retrieved from https:// blogs.mcafee.com/business/security-connected/tackling-the-botnet-threat

Tsai, C.-R., Gligor, V. D., & Chandersekaran, C. S. (1990). On the identification of covert storage channels in secure systems. *Software Engineering, IEEE Transactions on*, *16*(6), 569–580.

Tyagi, A. K., & Aghila, G. (2011). A wide scale survey on botnet. *International Journal of Computer Applications*, *34*(9), 9–22.

van Rijswijk-Deij, R., Sperotto, A., & Pras, A. (2014). Dnssec and its potential for ddos attacks: a comprehensive measurement study. In *Proceedings of the 2014 conference on internet measurement* (pp. 449–460).

Vixie, P. (1995). Dns and bind security issues. In *Usenix security.*

Vixie, P. (1999). Extension mechanisms for dns (edns0).

Wayner, P. (1997). *Private domain register upsets internic again.* Retrieved from https://partners.nytimes.com/library/cyber/week/072297/internic.html

Weimer, F. (2005). Passive dns replication. In *First conference on computer security incident* (p. 98).

Wijesinghe, U., Tupakula, U., & Varadharajan, V. (2015). An enhanced model for network flow based botnet detection. In *Proceedings of the 38th australasian computer science conference (acsc 2015)* (Vol. 27, p. 30).

Zhang, H., Papadopoulos, C., & Massey, D. (2013). Detecting encrypted botnet traffic. In *Computer communications workshops (infocom wkshps), 2013 ieee conference on* (pp. 163–168).

APPENDICES

Appendix A. Covert Channel Documentation

A.1 Covert Channel Whois Lookups

The transfer of data is demonstrated in the following series of Whois lookups of the DNSSEC signed domain frankfrank.org. The SOA record has a “serial" field, and this value changes with each lookup. In messages 1 - 6, if there are fewer than 10 digits, the differing number of digits would be added in leading zeros. The final message (number 7) requires a single leading zero. When converted from binary to text, these digits spell out the string “P@$$w0rd."

```
OrgAbuseRef:    https://whois.arin.net/rest/poc/ABUSE2202-ARIN

OrgTechHandle: HCM2-ARIN
OrgTechName:   hostmaster for Cinergy Metronet
OrgTechPhone:  +1-877-407-3224
OrgTechEmail:  CMN-hostmaster@metronetinc.com
OrgTechRef:    https://whois.arin.net/rest/poc/HCM2-ARIN

OrgTechHandle: CAMPB195-ARIN
OrgTechName:   Campbell, John
OrgTechPhone:  +1-913-754-3330
OrgTechEmail:  campbell@qservicesco.com
OrgTechRef:    https://whois.arin.net/rest/poc/CAMPB195-ARIN
```

DNS records

name	class	type	data		time to live
frankfrank.org	IN	SOA	server:	bender.frankfrank.org	604800s (7.00:00:00)
			email:	admin@frankfrank.org.frankfrank.org	
			serial:	101000001	
			refresh:	604800	
			retry:	86400	
			expire:	2419200	
			minimum ttl:	604800	
frankfrank.org	IN	RRSIG	type covered:	SOA (6)	604800s (7.00:00:00)
			algorithm:	RSASHA1-NSEC3-SHA1 (7)	
			labels:	2	
			original ttl:	604800 (7.00:00:00)	
			signature expiration:	2016-08-16 05:55:47Z	
			signature inception:	2016-07-17 05:55:47Z	
			key tag:	24884	
			signer's name:	frankfrank.org	
			signature: (2048 bits)	355244BAC7366E87A3429D1D4C6CED73 D7F66655382C1CA38D5851BE36FF947E F39762176B760452EDBED8A3A59CAC16 D549E676918A3868DBB4958F59E6D2A4 E4AC090837391492342 1BD5802EA08A0 51D26CB135FDE7ABC47E2439855E5CD3 41DDB604E31553CAABF3C7599393CC03 8F635A60FA4AB865F13FD9770500619C 855FCC6EFB176C8066E519E2C9147807 B7AD976061EFABCFF8F25DE89BA00F4D DCA9079ACB9314E2AF8F29052D1B6E5A 8C76132E4DBA5D9B89E5207D2FA61463 2A20DCED0EE31FCB0F0B3FB4CF72846E B2A8B901C8927A29DF79056FCD74024D C86A7C580755C70E223CAF7915EF8FE3 B5452537D8563E76BFA695CDA6AC85BA	
frankfrank.org	IN	NS	frank-cnc.frankfrank.org		604800s (7.00:00:00)

Figure A.1.: Whois Lookup 1 - The first 10 digits are transferred

```
OrgAbuseRef:    https://whois.arin.net/rest/poc/ABUSE2202-ARIN

OrgTechHandle: HCM2-ARIN
OrgTechName:   hostmaster for Cinergy Metronet
OrgTechPhone:  +1-877-407-3224
OrgTechEmail:  CMN-hostmaster@metronetinc.com
OrgTechRef:    https://whois.arin.net/rest/poc/HCM2-ARIN

OrgTechHandle: CAMPB195-ARIN
OrgTechName:   Campbell, John
OrgTechPhone:  +1-913-754-3330
OrgTechEmail:  campbell@qservicesco.com
OrgTechRef:    https://whois.arin.net/rest/poc/CAMPB195-ARIN
```

DNS records

name	class	type	data		time to live
frankfrank.org	IN	SOA	server:	bender.frankfrank.org	604800s (7.00:00:00)
			email:	admin@frankfrank.org.frankfrank.org	
			serial:	10	
			refresh:	604800	
			retry:	86400	
			expire:	2419200	
			minimum ttl:	604800	
frankfrank.org	IN	RRSIG	type covered:	SOA (6)	604800s (7.00:00:00)
			algorithm:	RSASHA1-NSEC3-SHA1 (7)	
			labels:	2	
			original ttl:	604800 (7.00:00:00)	
			signature expiration:	2016-08-16 06:07:40Z	
			signature inception:	2016-07-17 06:07:40Z	
			key tag:	24884	
			signer's name:	frankfrank.org	
			signature: (2048 bits)	3BCF49FF023107C4BE55844E7EE58B74 2534F35CD3BB8AF3BDE782ABABC81C38 9786902A6A1E30C06A3BD4D94721479E 2BE02B939F7C909B6086959441353306 CE3953CF6A5B0913750D0B6E4E1B0F64 6771CE86414B03794668FCA8853788CE 25141BDB1ADC7AE3F323354A9191011B A2A353389CB147DA5C3C950403BC2525 A6DB2464CD765F9FF9C61FC7F6CC4D6F 1B6B9ABAFD1BD84A9ED5C8F9CC28F1A5 F5226B0B5D3A9469466EC707F74B0D7E 946F9E48AE78947A38DDBCDD213E3ABD FCCA8394166977E27AEA594C847AE209 662FA6A216776EA47383A8917A0D76F7 07D10EA9EC9F0DDB524F74CD84B3F873 CF746EF23AE265B96AD5A6D93DBA3C20	
frankfrank.org	IN	NS	frank-cnc.frankfrank.org		604800s (7.00:00:00)

Figure A.2.: Whois Lookup 2 - The leading zeroes are removed

```
OrgAbuseRef:    https://whois.arin.net/rest/poc/ABUSE2202-ARIN

OrgTechHandle: HCM2-ARIN
OrgTechName:   hostmaster for Cinergy Metronet
OrgTechPhone:  +1-877-407-3224
OrgTechEmail:  CMN-hostmaster@metronetinc.com
OrgTechRef:    https://whois.arin.net/rest/poc/HCM2-ARIN

OrgTechHandle: CAMPB195-ARIN
OrgTechName:   Campbell, John
OrgTechPhone:  +1-913-754-3330
OrgTechEmail:  campbell@qservicesco.com
OrgTechRef:    https://whois.arin.net/rest/poc/CAMPB195-ARIN
```

DNS records

name	class	type	data		time to live
frankfrank.org	IN	SOA	server:	bender.frankfrank.org	604800s (7.00:00:00)
			email:	admin@frankfrank.org.frankfrank.org	
			serial:	100001001	
			refresh:	604800	
			retry:	86400	
			expire:	2419200	
			minimum ttl:	604800	
frankfrank.org	IN	RRSIG	type covered:	SOA (6)	604800s (7.00:00:00)
			algorithm:	RSASHA1-NSEC3-SHA1 (7)	
			labels:	2	
			original ttl:	604800 (7.00:00:00)	
			signature expiration:	2016-08-16 06:10:33Z	
			signature inception:	2016-07-17 06:10:33Z	
			key tag:	24884	
			signer's name:	frankfrank.org	
			signature: (2048 bits)	6ED7EA2ADC639C55DE228439722E82CE 832210962BAF13FB44936A7399982AF8 A6AFA0799727910DA19385E7025DF0BD F3B4D5EB926FF3FFC8CF7DB8E1B8729A 34B2B9F7300E1648B769F0157D5BA211 6C994EC2E10270C2D1BDF10B9FEC68A1 F3E95380AEEA7552C5A6E84502E10562 855ABA045472614842DB6E50718FABB6 21E4E6902875B18A6BE014AC5695D1CD A518B8520103DAC9D6EB42F1C8560420 9D67D2628A7610E3D0B8C00769899E14 D2C27FB7E237C0898711BCC59A53E3D6 D1A498B52BA4393303AAA61CD5CA4EC8 0FB9F51D94DAE0F553DF0369AD39D86A 7DC423916614 1D9396E8C03DE27E979B 5ED2C79541A2157A7EAAE3A74AF9CE3D	
frankfrank.org	IN	NS	bender.frankfrank.org		604800s (7.00:00:00)

Figure A.3.: Whois Lookup 3

```
OrgAbuseRef:    https://whois.arin.net/rest/poc/ABUSE2202-ARIN

OrgTechHandle: HCM2-ARIN
OrgTechName:   hostmaster for Cinergy Metronet
OrgTechPhone:  +1-877-407-3224
OrgTechEmail:  CMN-hostmaster@metronetinc.com
OrgTechRef:    https://whois.arin.net/rest/poc/HCM2-ARIN

OrgTechHandle: CAMPB195-ARIN
OrgTechName:   Campbell, John
OrgTechPhone:  +1-913-754-3330
OrgTechEmail:  campbell@qservicesco.com
OrgTechRef:    https://whois.arin.net/rest/poc/CAMPB195-ARIN
```

DNS records

name	class	type	data		time to live
frankfrank.org	IN	SOA	server:	bender.frankfrank.org	604800s (7.00:00:00)
			email:	admin@frankfrank.org.frankfrank.org	
			serial:	1110111	
			refresh:	604800	
			retry:	86400	
			expire:	2419200	
			minimum ttl:	604800	
frankfrank.org	IN	RRSIG	type covered:	SOA (6)	604800s (7.00:00:00)
			algorithm:	RSASHA1-NSEC3-SHA1 (7)	
			labels:	2	
			original ttl:	604800 (7.00:00:00)	
			signature expiration:	2016-08-16 06:12:51Z	
			signature inception:	2016-07-17 06:12:51Z	
			key tag:	24884	
			signer's name:	frankfrank.org	
			signature: (2048 bits)	50170D8081A68259D64431B734F30349 5F643422D817B964072BFD0F3D6C6F38 9D7CAC85884E6DEABBBE2D60474A79E8 E4082D5F7FF6E04CD9D4246D44DABAF5 858DB9C29F534E0B10694A2D8BB0600B 2D3D58E334C82072F96EA94103CF8E70 440AC1A1BC89F41AEF5D85BA71BB8A7B 53D3CA8A56EC74175E9562D751A276AA 3BC6BD621347B4DC8F33B7AA51C2FE0B B7FE31B69ADBF94A6A04CBA166A9076F6 5176907F1029B997F0E03409D57D9FE7 BB9CB9AE52670814EA7A586385F0A2F4 2F0A42577B58FB859910BAB40A3BF97D 3FCC6905D285C1B5C6AFF80D672725E4 AEAC4E9FA2B2D77DB4A2D624B0C58652 AD0DEA0E3660E2A018A3056D47C6D72B	
frankfrank.org	IN	NS	bender.frankfrank.org		604800s (7.00:00:00)

Figure A.4.: Whois Lookup 4

```
OrgAbuseRef:    https://whois.arin.net/rest/poc/ABUSE2202-ARIN

OrgTechHandle: HCM2-ARIN
OrgTechName:   hostmaster for Cinergy Metronet
OrgTechPhone:  +1-877-407-3224
OrgTechEmail:  CMN-hostmaster@metronetinc.com
OrgTechRef:    https://whois.arin.net/rest/poc/HCM2-ARIN

OrgTechHandle: CAMPB195-ARIN
OrgTechName:   Campbell, John
OrgTechPhone:  +1-913-754-3330
OrgTechEmail:  campbell@qservicesco.com
OrgTechRef:    https://whois.arin.net/rest/poc/CAMPB195-ARIN
```

DNS records

name	class	type	data		time to live
frankfrank.org	IN	SOA	server:	bender.frankfrank.org	604800s (7.00:00:00)
			email:	admin@frankfrank.org.frankfrank.org	
			serial:	11000001	
			refresh:	604800	
			retry:	86400	
			expire:	2419200	
			minimum ttl:	604800	
frankfrank.org	IN	RRSIG	type covered:	SOA (6)	604800s (7.00:00:00)
			algorithm:	RSASHA1-NSEC3-SHA1 (7)	
			labels:	2	
			original ttl:	604800 (7.00:00:00)	
			signature expiration:	2016-08-16 06:14:22Z	
			signature inception:	2016-07-17 06:14:22Z	
			key tag:	24884	
			signer's name:	frankfrank.org	
			signature: (2048 bits)	0C73D5426A865195B9E5DA5FA86403C7 9D44B926D2825C4AF8EA5E2FC94BB4CA 0930250BAAF96D27FA9EFC291FFD9C8E DAAB60A2C7031715B182DB114BEDCA8F D3006F2E2E176862EE8C340CA2B5E945 76288E96D7B48C85BBDE26470F10E9B5 84A5A9FFF28E3A1120065D4818C00E56 617BC181CBF8E917F8552E8D49322E94 8E6E3563C40ECECCF6CF91127CE62D49 66E2E10EE461AE30050FFEA6F25521A8 58302C2C4881E0802E20A012A6270C35 F93BE1DCAE0C01BC398B3D79C6E82CC7 5B246066E05117D3DB8004C292851B24 B237A4A0CEF7101A994397A1A312F682 4AC9B358891558A7D14B42397D471B5B EEC147E0771F8CC0EF28BA6AFB4E2BC8	
frankfrank.org	IN	NS	bender.frankfrank.org		604800s (7.00:00:00)

Figure A.5.: Whois Lookup 5

```
OrgAbuseRef:    https://whois.arin.net/rest/poc/ABUSE2202-ARIN

OrgTechHandle: HCM2-ARIN
OrgTechName:   hostmaster for Cinergy Metronet
OrgTechPhone:  +1-877-407-3224
OrgTechEmail:  CMN-hostmaster@metronetinc.com
OrgTechRef:    https://whois.arin.net/rest/poc/HCM2-ARIN

OrgTechHandle: CAMPB195-ARIN
OrgTechName:   Campbell, John
OrgTechPhone:  +1-913-754-3330
OrgTechEmail:  campbell@qservicesco.com
OrgTechRef:    https://whois.arin.net/rest/poc/CAMPB195-ARIN
```

DNS records

name	class	type	data		time to live
frankfrank.org	IN	SOA	server:	bender.frankfrank.org	604800s (7.00:00:00)
			email:	admin@frankfrank.org.frankfrank.org	
			serial:	1100100110	
			refresh:	604800	
			retry:	86400	
			expire:	2419200	
			minimum ttl:	604800	
frankfrank.org	IN	RRSIG	type covered:	SOA (6)	604800s (7.00:00:00)
			algorithm:	RSASHA1-NSEC3-SHA1 (7)	
			labels:	2	
			original ttl:	604800 (7.00:00:00)	
			signature expiration:	2016-08-16 06:16:39Z	
			signature inception:	2016-07-17 06:16:39Z	
			key tag:	24884	
			signer's name:	frankfrank.org	
			signature: (2048 bits)	66308EF6AD8A3935B4103A5FC4927BB5 D8D0268C38DD21DFFEBF97633CD8D565 8663FA47151912F75703170EEF336347 C46CC3E3CA9390CDF3AB8C10514A5431 9669170D1B3340DF08E4B2F4083658D9 F66F2E46A6986F83829DC87970B5CC79 6A55393E63CA0B2EA4DA111756F9E9AD 60E14B6A32C76C20F99267824 6DC37BB E0560DE06D07899538633D5FC6C1EBD5 2FFD536D4F66BF7C22DE2D3878BFB511 B23BF368FA232ED6D1E02B8A2387E45B 5FEC1557F1C40909D21EE97A8D64EAA7 E4D561C238431B3E52FA58EDD3D159DA C30BCC139E1CF77F4293B4715D69E402 1027A10E7FBDC83C210D870E98B4446C 12EF32E62A9C93C3CC1BE8301FABBC3C	
frankfrank.org	IN	NS	frank-cnc.frankfrank.org		604800s (7.00:00:00)

Figure A.6.: Whois Lookup 6

```
OrgAbuseRef:    https://whois.arin.net/rest/poc/ABUSE2202-ARIN

OrgTechHandle: HCM2-ARIN
OrgTechName:   hostmaster for Cinergy Metronet
OrgTechPhone:  +1-877-407-3224
OrgTechEmail:  CMN-hostmaster@metronetinc.com
OrgTechRef:    https://whois.arin.net/rest/poc/HCM2-ARIN

OrgTechHandle: CAMPB195-ARIN
OrgTechName:   Campbell, John
OrgTechPhone:  +1-913-754-3330
OrgTechEmail:  campbell@qservicesco.com
OrgTechRef:    https://whois.arin.net/rest/poc/CAMPB195-ARIN
```

DNS records

name	class	type	data		time to live
frankfrank.org	IN	SOA	server:	bender.frankfrank.org	604800s (7.00:00:00)
			email:	admin@frankfrank.org.frankfrank.org	
			serial:	100	
			refresh:	604800	
			retry:	86400	
			expire:	2419200	
			minimum ttl:	604800	
frankfrank.org	IN	RRSIG	type covered:	SOA (6)	604800s (7.00:00:00)
			algorithm:	RSASHA1-NSEC3-SHA1 (7)	
			labels:	2	
			original ttl:	604800 (7.00:00:00)	
			signature expiration:	2016-08-16 06:18:44Z	
			signature inception:	2016-07-17 06:18:44Z	
			key tag:	24884	
			signer's name:	frankfrank.org	
			signature: (2048 bits)	8AE34C7D8478A565D5EA7FF1F16F47D6 8C4D3B3FAB8D285E24EB1DB15258310E BF357032B4F63FA90DD8A3B72B174B9E B3B397DE506A472339DCF84D980713B1 3849F567C51A704FC2188B56FC69DFA7 B7E86A079741A336C4368D1BA0523976 439E5D9CF9C388F6445FC357793B94FC 2D8BEFC2F8F46F7FFB600EC637F6E6F3 8791F8E2D3963464D33DC188622E95D1 652356878CE19B0E947E05B081EDCCC6 4B9237E844BDECD52B3F951451887AF7 9D4C684FE68826AF4A427C19BECA92FD FE1424A4B033871C60663C7D1198369B 639AC9C5B819316EC679CFCD30E8D3B6 EC524CBEA6149D494F0FC3D680A570D1 ACF2E0DF648805FE39E56B67EF950FFE	
frankfrank.org	IN	NS	frank-cnc.frankfrank.org		604800s (7.00:00:00)

Figure A.7.: Whois Lookup 7 - The remaining 4 bits are sent

Appendix B. Master Nameserver System Logs

B.1 Master Syslogs

The syslogs record certain changes to the BIND service and its associated files. In the following figures, the service can be seen to be loaded, reloaded, the zone file changed, and notifies sent. In figure B.2, the first attempt at decrementing the SOA serial field is shown. It is recorded in the syslog as "zone serial has gone backwards."Figures B.3, B.4, and B.5 show the zone updates that led to the data transfer. Figure B.7 shows the update to a serial incremented from the previous version known by the slave - with an accompanying zone transfer demonstrated in Appendix C, Figure C.3.

```
Jul 17 00:48:07 bender named[16638]: received control channel command 'reload'
Jul 17 00:48:07 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 00:48:07 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 00:48:07 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 00:48:07 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 00:48:08 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 00:48:08 bender named[16638]: using built-in DLV key for view _default
Jul 17 00:48:08 bender named[16638]: reloading configuration succeeded
Jul 17 00:48:08 bender named[16638]: reloading zones succeeded
Jul 17 00:48:08 bender named[16638]: all zones loaded
Jul 17 00:48:08 bender named[16638]: running
Jul 17 00:48:08 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 00:48:08 bender named[16638]: zone frankfrank.org/IN: loaded serial 201607114 (DNSSEC
signed)
Jul 17 00:48:08 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 201607114)
Jul 17 00:52:18 bender named[16638]: received control channel command 'reload'
Jul 17 00:52:18 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 00:52:18 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 00:52:18 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 00:52:18 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 00:52:18 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 00:52:18 bender named[16638]: using built-in DLV key for view _default
Jul 17 00:52:18 bender named[16638]: reloading configuration succeeded
Jul 17 00:52:18 bender named[16638]: reloading zones succeeded
Jul 17 00:52:18 bender named[16638]: all zones loaded
Jul 17 00:52:18 bender named[16638]: running
Jul 17 00:52:18 bender named[16638]: zone frankfrank.org/IN: zone serial (201607114) unchanged.
zone may fail to transfer to slaves.
Jul 17 00:52:18 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 00:52:18 bender named[16638]: zone frankfrank.org/IN: loaded serial 201607114 (DNSSEC
signed)
Jul 17 00:52:18 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 201607114)
Jul 17 00:58:42 bender named[16638]: received control channel command 'reload'
Jul 17 00:58:42 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 00:58:42 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 00:58:42 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 00:58:42 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 00:58:42 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 00:58:42 bender named[16638]: using built-in DLV key for view _default
Jul 17 00:58:42 bender named[16638]: reloading configuration succeeded
Jul 17 00:58:42 bender named[16638]: reloading zones succeeded
Jul 17 00:58:42 bender named[16638]: all zones loaded
Jul 17 00:58:42 bender named[16638]: running
Jul 17 00:58:42 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 00:58:42 bender named[16638]: zone frankfrank.org/IN: loaded serial 201607172 (DNSSEC
signed)
Jul 17 00:58:42 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 201607172)
--------------------------------------------------------------------------------------------
-----------------
Jul 17 02:36:29 bender named[16638]: received control channel command 'reload'
Jul 17 02:36:29 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 02:36:29 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 02:36:29 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 02:36:29 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 02:36:29 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 02:36:29 bender named[16638]: using built-in DLV key for view _default
Jul 17 02:36:29 bender named[16638]: reloading configuration succeeded
Jul 17 02:36:29 bender named[16638]: reloading zones succeeded
Jul 17 02:36:29 bender named[16638]: all zones loaded
```

-1-

Figure B.1.: System Log 1 for Master Nameserver

```
Jul 17 02:36:29 bender named[16638]: running
--------------------------------------------------------------------------------
------------------
Jul 17 02:41:34 bender named[16638]: received control channel command 'reload'
Jul 17 02:41:34 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 02:41:34 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 02:41:34 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 02:41:34 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 02:41:34 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 02:41:34 bender named[16638]: using built-in DLV key for view _default
Jul 17 02:41:34 bender named[16638]: reloading configuration succeeded
Jul 17 02:41:34 bender named[16638]: reloading zones succeeded
Jul 17 02:41:34 bender named[16638]: all zones loaded
Jul 17 02:41:34 bender named[16638]: running
Jul 17 02:41:34 bender named[16638]: zone frankfrank.org/IN: zone serial (201607173) unchanged.
zone may fail to transfer to slaves.
Jul 17 02:41:34 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 02:41:34 bender named[16638]: zone frankfrank.org/IN: loaded serial 201607173 (DNSSEC
signed)
Jul 17 02:41:34 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 201607173)
Jul 17 02:45:37 bender named[16638]: received control channel command 'reload'
Jul 17 02:45:37 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 02:45:37 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 02:45:37 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 02:45:37 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 02:45:37 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 02:45:37 bender named[16638]: using built-in DLV key for view _default
Jul 17 02:45:37 bender named[16638]: reloading configuration succeeded
Jul 17 02:45:37 bender named[16638]: reloading zones succeeded
Jul 17 02:45:37 bender named[16638]: all zones loaded
Jul 17 02:45:37 bender named[16638]: running
Jul 17 02:45:37 bender named[16638]: zone frankfrank.org/IN: zone serial (201607171/201607173)
has gone backwards
Jul 17 02:45:37 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 02:45:37 bender named[16638]: zone frankfrank.org/IN: loaded serial 201607171 (DNSSEC
signed)
Jul 17 02:45:37 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 201607171)
Jul 17 02:49:14 bender named[16638]: received control channel command 'reload'
Jul 17 02:49:14 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 02:49:14 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 02:49:14 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 02:49:14 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 02:49:14 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 02:49:14 bender named[16638]: using built-in DLV key for view _default
Jul 17 02:49:14 bender named[16638]: reloading configuration succeeded
Jul 17 02:49:14 bender named[16638]: reloading zones succeeded
Jul 17 02:49:14 bender named[16638]: all zones loaded
Jul 17 02:49:14 bender named[16638]: running
Jul 17 02:49:14 bender named[16638]: zone frankfrank.org/IN: zone serial (123456124/201607171)
has gone backwards
Jul 17 02:49:14 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 02:49:14 bender named[16638]: zone frankfrank.org/IN: loaded serial 123456124 (DNSSEC
signed)
Jul 17 02:49:14 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 123456124)
Jul 17 02:56:08 bender named[16638]: received control channel command 'reload'
Jul 17 02:56:08 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 02:56:08 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 02:56:08 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 02:56:08 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
```

-2-

Figure B.2.: System Log 2 for Master Nameserver

```
Jul 17 02:56:08 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 02:56:08 bender named[16638]: using built-in DLV key for view _default
Jul 17 02:56:08 bender named[16638]: reloading configuration succeeded
Jul 17 02:56:08 bender named[16638]: reloading zones succeeded
Jul 17 02:56:08 bender named[16638]: all zones loaded
Jul 17 02:56:08 bender named[16638]: running
Jul 17 02:56:08 bender named[16638]: zone frankfrank.org/IN: zone serial (101000001/123456124)
has gone backwards
Jul 17 02:56:08 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 02:56:08 bender named[16638]: zone frankfrank.org/IN: loaded serial 101000001 (DNSSEC
signed)
Jul 17 02:56:08 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 101000001)
Jul 17 02:58:53 bender named[16638]: received control channel command 'reload'
Jul 17 02:58:53 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 02:58:53 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 02:58:53 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 02:58:53 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 02:58:53 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 02:58:53 bender named[16638]: using built-in DLV key for view _default
Jul 17 02:58:53 bender named[16638]: reloading configuration succeeded
Jul 17 02:58:53 bender named[16638]: reloading zones succeeded
Jul 17 02:58:53 bender named[16638]: all zones loaded
Jul 17 02:58:53 bender named[16638]: running
Jul 17 02:58:53 bender named[16638]: zone frankfrank.org/IN: zone serial (101000001) unchanged.
zone may fail to transfer to slaves.
Jul 17 02:58:53 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 02:58:53 bender named[16638]: zone frankfrank.org/IN: loaded serial 101000001 (DNSSEC
signed)
Jul 17 02:58:53 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 101000001)
Jul 17 03:01:42 bender named[16638]: received control channel command 'reload'
Jul 17 03:01:42 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 03:01:42 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 03:01:42 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 03:01:42 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 03:01:42 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 03:01:42 bender named[16638]: using built-in DLV key for view _default
Jul 17 03:01:42 bender named[16638]: reloading configuration succeeded
Jul 17 03:01:42 bender named[16638]: reloading zones succeeded
Jul 17 03:01:42 bender named[16638]: all zones loaded
Jul 17 03:01:42 bender named[16638]: running
Jul 17 03:01:42 bender named[16638]: zone frankfrank.org/IN: zone serial (101000001) unchanged.
zone may fail to transfer to slaves.
Jul 17 03:01:42 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 03:01:42 bender named[16638]: zone frankfrank.org/IN: loaded serial 101000001 (DNSSEC
signed)
Jul 17 03:01:42 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 101000001)
-------------------------------------------------------------------------------------------
-----------------------------------
Jul 17 03:07:46 bender named[16638]: received control channel command 'reload'
Jul 17 03:07:46 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 03:07:46 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 03:07:46 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 03:07:46 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 03:07:46 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 03:07:46 bender named[16638]: using built-in DLV key for view _default
Jul 17 03:07:46 bender named[16638]: reloading configuration succeeded
Jul 17 03:07:46 bender named[16638]: reloading zones succeeded
Jul 17 03:07:46 bender named[16638]: all zones loaded
Jul 17 03:07:46 bender named[16638]: running
Jul 17 03:07:46 bender named[16638]: zone frankfrank.org/IN: zone serial (10/101000001) has
```


Figure B.3.: System Log 3 for Master Nameserver

```
gone backwards
Jul 17 03:07:46 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 03:07:46 bender named[16638]: zone frankfrank.org/IN: loaded serial 10 (DNSSEC signed)
Jul 17 03:07:46 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 10)
------------------------------------------------------------------------------------------
----------------------
Jul 17 03:10:40 bender named[16638]: received control channel command 'reload'
Jul 17 03:10:40 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 03:10:40 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 03:10:40 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 03:10:40 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 03:10:40 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 03:10:40 bender named[16638]: using built-in DLV key for view _default
Jul 17 03:10:40 bender named[16638]: reloading configuration succeeded
Jul 17 03:10:40 bender named[16638]: reloading zones succeeded
Jul 17 03:10:40 bender named[16638]: all zones loaded
Jul 17 03:10:40 bender named[16638]: running
Jul 17 03:10:40 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 03:10:40 bender named[16638]: zone frankfrank.org/IN: loaded serial 100001001 (DNSSEC
signed)
Jul 17 03:10:40 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 100001001)
Jul 17 03:12:56 bender named[16638]: received control channel command 'reload'
Jul 17 03:12:56 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 03:12:56 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 03:12:56 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 03:12:56 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 03:12:56 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 03:12:56 bender named[16638]: using built-in DLV key for view _default
Jul 17 03:12:56 bender named[16638]: reloading configuration succeeded
Jul 17 03:12:56 bender named[16638]: reloading zones succeeded
Jul 17 03:12:56 bender named[16638]: all zones loaded
Jul 17 03:12:56 bender named[16638]: running
Jul 17 03:12:56 bender named[16638]: zone frankfrank.org/IN: zone serial (1110111/100001001)
has gone backwards
Jul 17 03:12:56 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 03:12:56 bender named[16638]: zone frankfrank.org/IN: loaded serial 1110111 (DNSSEC
signed)
Jul 17 03:12:56 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 1110111)
Jul 17 03:14:31 bender named[16638]: received control channel command 'reload'
Jul 17 03:14:31 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 03:14:31 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 03:14:31 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 03:14:31 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 03:14:31 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 03:14:31 bender named[16638]: using built-in DLV key for view _default
Jul 17 03:14:31 bender named[16638]: reloading configuration succeeded
Jul 17 03:14:31 bender named[16638]: reloading zones succeeded
Jul 17 03:14:31 bender named[16638]: all zones loaded
Jul 17 03:14:31 bender named[16638]: running
Jul 17 03:14:31 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 03:14:31 bender named[16638]: zone frankfrank.org/IN: loaded serial 11000001 (DNSSEC
signed)
Jul 17 03:14:31 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 11000001)
----------------------------------

Jul 17 03:16:51 bender named[16638]: received control channel command 'reload'
Jul 17 03:16:51 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 03:16:51 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'

-4-
```

Figure B.4.: System Log 4 for Master Nameserver

```
Jul 17 03:16:51 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 03:16:51 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 03:16:51 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 03:16:51 bender named[16638]: using built-in DLV key for view _default
Jul 17 03:16:51 bender named[16638]: reloading configuration succeeded
Jul 17 03:16:51 bender named[16638]: reloading zones succeeded
Jul 17 03:16:51 bender named[16638]: all zones loaded
Jul 17 03:16:51 bender named[16638]: running
Jul 17 03:16:51 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 03:16:51 bender named[16638]: zone frankfrank.org/IN: loaded serial 1100100110 (DNSSEC
signed)
Jul 17 03:16:51 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 1100100110)
Jul 17 03:17:01 bender CRON[19407]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Jul 17 03:18:48 bender named[16638]: received control channel command 'reload'
Jul 17 03:18:48 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 03:18:48 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 03:18:48 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 03:18:48 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 03:18:48 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 03:18:48 bender named[16638]: using built-in DLV key for view _default
Jul 17 03:18:48 bender named[16638]: reloading configuration succeeded
Jul 17 03:18:48 bender named[16638]: reloading zones succeeded
Jul 17 03:18:48 bender named[16638]: all zones loaded
Jul 17 03:18:48 bender named[16638]: running
Jul 17 03:18:49 bender named[16638]: zone frankfrank.org/IN: zone serial (100/1100100110) has
gone backwards
Jul 17 03:18:49 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 03:18:49 bender named[16638]: zone frankfrank.org/IN: loaded serial 100 (DNSSEC signed)
Jul 17 03:18:49 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 100)
--------------------------------------------------------------------------------------------
----------------------------------

Jul 17 03:42:39 bender named[16638]: received control channel command 'reload'
Jul 17 03:42:39 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 03:42:39 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 03:42:39 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 03:42:39 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 03:42:39 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 03:42:39 bender named[16638]: using built-in DLV key for view _default
Jul 17 03:42:39 bender named[16638]: reloading configuration succeeded
Jul 17 03:42:39 bender named[16638]: reloading zones succeeded
Jul 17 03:42:39 bender named[16638]: all zones loaded
Jul 17 03:42:39 bender named[16638]: running
Jul 17 03:43:26 bender named[16638]: received control channel command 'reload'
Jul 17 03:43:26 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 03:43:26 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 03:43:26 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 03:43:26 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 03:43:26 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 03:43:26 bender named[16638]: using built-in DLV key for view _default
Jul 17 03:43:26 bender named[16638]: reloading configuration succeeded
Jul 17 03:43:26 bender named[16638]: reloading zones succeeded
Jul 17 03:43:26 bender named[16638]: all zones loaded
Jul 17 03:43:26 bender named[16638]: running
Jul 17 03:43:26 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 03:43:26 bender named[16638]: zone frankfrank.org/IN: loaded serial 201607174 (DNSSEC
signed)
Jul 17 03:43:26 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 201607174)
--------------------------------------------------------------------------------------------
------------

-5-
```

Figure B.5.: System Log 5 for Master Nameserver

```
Jul 17 05:22:45 bender named[16638]: received control channel command 'reload'
Jul 17 05:22:45 bender named[16638]: loading configuration from '/etc/bind/named.conf'
Jul 17 05:22:45 bender named[16638]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 05:22:45 bender named[16638]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 05:22:45 bender named[16638]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 05:22:45 bender named[16638]: sizing zone task pool based on 6 zones
Jul 17 05:22:45 bender named[16638]: using built-in DLV key for view _default
Jul 17 05:22:45 bender named[16638]: reloading configuration succeeded
Jul 17 05:22:45 bender named[16638]: reloading zones succeeded
Jul 17 05:22:45 bender named[16638]: all zones loaded
Jul 17 05:22:45 bender named[16638]: running
Jul 17 05:22:45 bender named[16638]: zone frankfrank.org/IN: zone serial (201607174) unchanged.
zone may fail to transfer to slaves.
Jul 17 05:22:45 bender named[16638]: zone frankfrank.org/IN: sig-re-signing-interval less than
3 * refresh.
Jul 17 05:22:45 bender named[16638]: zone frankfrank.org/IN: loaded serial 201607174 (DNSSEC
signed)
Jul 17 05:22:45 bender named[16638]: zone frankfrank.org/IN: sending notifies (serial 201607174)

------------------------------------------------------------------------------------------------
----------------------------------------
```

-6-

Figure B.6.: System Log 6 for Master Nameserver

Appendix C. Slave Nameserver System Logs

The syslogs record certain changes to the BIND service and its associated files. In the following figures, the service can be seen to be loaded, reloaded, the zone file changed, and notifies sent. In the slave nameserver, the zone transfer can also be seen to begin and end. No zone transfers were recorded, nor was any communication from the master nameserver, during the time the data was being transferred over the covert channel.

C.1 Slave Nameserver Syslogs

```
-----------------------
Jul 12 07:19:04 frank-cnc named[1065]: zone frankfrank.org/IN: Transfer started.
Jul 12 07:19:04 frank-cnc named[1065]: transfer of 'frankfrank.org/IN' from 192.168.1.97#53:
connected using 192.168.65.133#49694
Jul 12 07:19:04 frank-cnc named[1065]: zone frankfrank.org/IN: transferred serial 201607111
Jul 12 07:19:04 frank-cnc named[1065]: transfer of 'frankfrank.org/IN' from 192.168.1.97#53:
Transfer completed: 1 messages, 28 records, 5602 bytes, 0.010 secs (560200 bytes/sec)
Jul 12 07:19:04 frank-cnc named[1065]: zone frankfrank.org/IN: sending notifies (serial
201607111)
-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------
Jul 14 06:38:09 frank-cnc named[1063]: starting BIND 9.9.5-3ubuntu0.8-Ubuntu -u bind
Jul 14 06:38:09 frank-cnc named[1063]: built with '--prefix=/usr' '--mandir=/usr/share/man'
'--infodir=/usr/share/info' '--sysconfdir=/etc/bind' '--localstatedir=/var' '--enable-threads'
'--enable-largefile' '--with-libtool' '--enable-shared' '--enable-static' '--with-openssl=/usr'
'--with-gssapi=/usr' '--with-gnu-ld' '--with-geoip=/usr' '--with-atf=no' '--enable-ipv6'
'--enable-rrl' '--enable-filter-aaaa' 'CFLAGS=-fno-strict-aliasing -DDIG_SIGCHASE -O2'
Jul 14 06:38:09 frank-cnc named[1063]: ------------------------------------------------
Jul 14 06:38:09 frank-cnc named[1063]: BIND 9 is maintained by Internet Systems Consortium,
Jul 14 06:38:09 frank-cnc named[1063]: Inc. (ISC), a non-profit 501(c)(3) public-benefit
Jul 14 06:38:09 frank-cnc named[1063]: corporation.  Support and training for BIND 9 are
Jul 14 06:38:09 frank-cnc named[1063]: available at https://www.isc.org/support
Jul 14 06:38:09 frank-cnc named[1063]: ------------------------------------------------
Jul 14 06:38:09 frank-cnc named[1063]: adjusted limit on open files from 4096 to 1048576
Jul 14 06:38:09 frank-cnc named[1063]: found 1 CPU, using 1 worker thread
Jul 14 06:38:09 frank-cnc named[1063]: using 1 UDP listener per interface
Jul 14 06:38:09 frank-cnc named[1063]: using up to 4096 sockets
Jul 14 06:38:09 frank-cnc named[1063]: loading configuration from '/etc/bind/named.conf'
Jul 14 06:38:09 frank-cnc named[1063]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 14 06:38:09 frank-cnc named[1063]: using default UDP/IPv4 port range: [1024, 65535]
Jul 14 06:38:09 frank-cnc named[1063]: using default UDP/IPv6 port range: [1024, 65535]
Jul 14 06:38:09 frank-cnc named[1063]: listening on IPv6 interfaces, port 53
Jul 14 06:38:09 frank-cnc named[1063]: listening on IPv4 interface lo, 127.0.0.1#53
Jul 14 06:38:09 frank-cnc named[1063]: listening on IPv4 interface eth0, 192.168.65.133#53
Jul 14 06:38:09 frank-cnc named[1063]: generating session key for dynamic DNS
Jul 14 06:38:09 frank-cnc named[1063]: sizing zone task pool based on 6 zones
Jul 14 06:38:09 frank-cnc named[1063]: using built-in DLV key for view _default
Jul 14 06:38:09 frank-cnc named[1063]: set up managed keys zone for view _default, file
'managed-keys.bind'
Jul 14 06:38:09 frank-cnc named[1063]: command channel listening on 127.0.0.1#953
Jul 14 06:38:09 frank-cnc named[1063]: command channel listening on ::1#953
Jul 14 06:38:09 frank-cnc named[1063]: managed-keys-zone: journal file is out of date: removing
journal file
Jul 14 06:38:09 frank-cnc named[1063]: managed-keys-zone: loaded serial 158
Jul 14 06:38:09 frank-cnc named[1063]: zone 0.in-addr.arpa/IN: loaded serial 1
Jul 14 06:38:09 frank-cnc named[1063]: zone 127.in-addr.arpa/IN: loaded serial 1
Jul 14 06:38:09 frank-cnc named[1063]: zone 255.in-addr.arpa/IN: loaded serial 1
Jul 14 06:38:09 frank-cnc named[1063]: zone localhost/IN: loaded serial 2
Jul 14 06:38:09 frank-cnc named[1063]: zone frankfrank.org/IN: loaded serial 201607111 (DNSSEC
signed)
Jul 14 06:38:09 frank-cnc named[1063]: all zones loaded
Jul 14 06:38:09 frank-cnc named[1063]: running
Jul 14 06:38:09 frank-cnc named[1063]: zone frankfrank.org/IN: sending notifies (serial
201607111)
-------------------------
Jul 15 08:32:21 frank-cnc named[1034]: starting BIND 9.9.5-3ubuntu0.8-Ubuntu -u bind
Jul 15 08:32:21 frank-cnc named[1034]: built with '--prefix=/usr' '--mandir=/usr/share/man'
'--infodir=/usr/share/info' '--sysconfdir=/etc/bind' '--localstatedir=/var' '--enable-threads'
'--enable-largefile' '--with-libtool' '--enable-shared' '--enable-static' '--with-openssl=/usr'
'--with-gssapi=/usr' '--with-gnu-ld' '--with-geoip=/usr' '--with-atf=no' '--enable-ipv6'
'--enable-rrl' '--enable-filter-aaaa' 'CFLAGS=-fno-strict-aliasing -DDIG_SIGCHASE -O2'
Jul 15 08:32:21 frank-cnc named[1034]: ------------------------------------------------
Jul 15 08:32:21 frank-cnc named[1034]: BIND 9 is maintained by Internet Systems Consortium,
Jul 15 08:32:21 frank-cnc named[1034]: Inc. (ISC), a non-profit 501(c)(3) public-benefit
```


Figure C.1.: System Log 1 for Slave Nameserver

```
Jul 15 08:32:21 frank-cnc named[1034]: corporation.  Support and training for BIND 9 are
Jul 15 08:32:21 frank-cnc named[1034]: available at https://www.isc.org/support
Jul 15 08:32:21 frank-cnc named[1034]: --------------------------------------------------
Jul 15 08:32:21 frank-cnc named[1034]: adjusted limit on open files from 4096 to 1048576
Jul 15 08:32:21 frank-cnc named[1034]: found 1 CPU, using 1 worker thread
Jul 15 08:32:21 frank-cnc named[1034]: using 1 UDP listener per interface
Jul 15 08:32:21 frank-cnc named[1034]: using up to 4096 sockets
Jul 15 08:32:21 frank-cnc named[1034]: loading configuration from '/etc/bind/named.conf'
Jul 15 08:32:21 frank-cnc named[1034]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 15 08:32:21 frank-cnc named[1034]: using default UDP/IPv4 port range: [1024, 65535]
Jul 15 08:32:21 frank-cnc named[1034]: using default UDP/IPv6 port range: [1024, 65535]
Jul 15 08:32:21 frank-cnc named[1034]: listening on IPv6 interfaces, port 53
Jul 15 08:32:21 frank-cnc named[1034]: listening on IPv4 interface lo, 127.0.0.1#53
Jul 15 08:32:21 frank-cnc named[1034]: listening on IPv4 interface eth0, 192.168.65.133#53
Jul 15 08:32:21 frank-cnc named[1034]: generating session key for dynamic DNS
Jul 15 08:32:21 frank-cnc named[1034]: sizing zone task pool based on 6 zones
Jul 15 08:32:21 frank-cnc named[1034]: using built-in DLV key for view _default
Jul 15 08:32:21 frank-cnc named[1034]: set up managed keys zone for view _default, file
'managed-keys.bind'
Jul 15 08:32:21 frank-cnc named[1034]: command channel listening on 127.0.0.1#953
Jul 15 08:32:21 frank-cnc named[1034]: command channel listening on ::1#953
Jul 15 08:32:21 frank-cnc named[1034]: managed-keys-zone: journal file is out of date: removing
journal file
Jul 15 08:32:21 frank-cnc named[1034]: managed-keys-zone: loaded serial 178
Jul 15 08:32:21 frank-cnc named[1034]: zone 0.in-addr.arpa/IN: loaded serial 1
Jul 15 08:32:21 frank-cnc named[1034]: zone 127.in-addr.arpa/IN: loaded serial 1
Jul 15 08:32:21 frank-cnc named[1034]: zone 255.in-addr.arpa/IN: loaded serial 1
Jul 15 08:32:21 frank-cnc named[1034]: zone localhost/IN: loaded serial 2
Jul 15 08:32:21 frank-cnc named[1034]: zone frankfrank.org/IN: loaded serial 201607111 (DNSSEC
signed)
Jul 15 08:32:21 frank-cnc named[1034]: all zones loaded
Jul 15 08:32:21 frank-cnc named[1034]: running
Jul 15 08:32:21 frank-cnc named[1034]: zone frankfrank.org/IN: sending notifies (serial
201607111)
--------------------------------------------------------------------------------------------
--------------------
Jul 17 00:48:19 frank-cnc named[1034]: received control channel command 'reload'
Jul 17 00:48:19 frank-cnc named[1034]: loading configuration from '/etc/bind/named.conf'
Jul 17 00:48:19 frank-cnc named[1034]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 00:48:19 frank-cnc named[1034]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 00:48:19 frank-cnc named[1034]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 00:48:19 frank-cnc named[1034]: sizing zone task pool based on 6 zones
Jul 17 00:48:19 frank-cnc named[1034]: using built-in DLV key for view _default
Jul 17 00:48:19 frank-cnc named[1034]: reloading configuration succeeded
Jul 17 00:48:19 frank-cnc named[1034]: reloading zones succeeded
Jul 17 00:48:19 frank-cnc named[1034]: all zones loaded
Jul 17 00:48:19 frank-cnc named[1034]: running
--------------------------------------------------------------------------------------------
--------------------
Jul 17 00:50:16 frank-cnc named[1034]: received control channel command 'reload'
Jul 17 00:50:16 frank-cnc named[1034]: loading configuration from '/etc/bind/named.conf'
Jul 17 00:50:16 frank-cnc named[1034]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 00:50:16 frank-cnc named[1034]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 00:50:16 frank-cnc named[1034]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 00:50:16 frank-cnc named[1034]: sizing zone task pool based on 6 zones
Jul 17 00:50:16 frank-cnc named[1034]: using built-in DLV key for view _default
Jul 17 00:50:16 frank-cnc named[1034]: reloading configuration succeeded
Jul 17 00:50:16 frank-cnc named[1034]: reloading zones succeeded
Jul 17 00:50:16 frank-cnc named[1034]: all zones loaded
Jul 17 00:50:16 frank-cnc named[1034]: running
Jul 17 00:56:10 frank-cnc named[1034]: received control channel command 'reload'
Jul 17 00:56:10 frank-cnc named[1034]: loading configuration from '/etc/bind/named.conf'
Jul 17 00:56:10 frank-cnc named[1034]: reading built-in trusted keys from file
```

-2-

Figure C.2.: System Log 2 for Slave Nameserver

```
'/etc/bind/bind.keys'
Jul 17 00:56:10 frank-cnc named[1034]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 00:56:10 frank-cnc named[1034]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 00:56:10 frank-cnc named[1034]: sizing zone task pool based on 6 zones
Jul 17 00:56:10 frank-cnc named[1034]: using built-in DLV key for view _default
Jul 17 00:56:10 frank-cnc named[1034]: reloading configuration succeeded
Jul 17 00:56:10 frank-cnc named[1034]: reloading zones succeeded
Jul 17 00:56:10 frank-cnc named[1034]: all zones loaded
Jul 17 00:56:10 frank-cnc named[1034]: running
Jul 17 00:59:14 frank-cnc named[1034]: received control channel command 'reload'
Jul 17 00:59:14 frank-cnc named[1034]: loading configuration from '/etc/bind/named.conf'
Jul 17 00:59:14 frank-cnc named[1034]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 00:59:14 frank-cnc named[1034]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 00:59:14 frank-cnc named[1034]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 00:59:14 frank-cnc named[1034]: sizing zone task pool based on 6 zones
Jul 17 00:59:14 frank-cnc named[1034]: using built-in DLV key for view _default
Jul 17 00:59:14 frank-cnc named[1034]: reloading configuration succeeded
Jul 17 00:59:14 frank-cnc named[1034]: reloading zones succeeded
Jul 17 00:59:14 frank-cnc named[1034]: all zones loaded
Jul 17 00:59:14 frank-cnc named[1034]: running
Jul 17 01:01:31 frank-cnc named[1034]: received control channel command 'reload'
Jul 17 01:01:31 frank-cnc named[1034]: loading configuration from '/etc/bind/named.conf'
Jul 17 01:01:31 frank-cnc named[1034]: reading built-in trusted keys from file
'/etc/bind/bind.keys'
Jul 17 01:01:31 frank-cnc named[1034]: using default UDP/IPv4 port range: [1024, 65535]
Jul 17 01:01:31 frank-cnc named[1034]: using default UDP/IPv6 port range: [1024, 65535]
Jul 17 01:01:31 frank-cnc named[1034]: sizing zone task pool based on 6 zones
Jul 17 01:01:31 frank-cnc named[1034]: using built-in DLV key for view _default
Jul 17 01:01:31 frank-cnc named[1034]: reloading configuration succeeded
Jul 17 01:01:31 frank-cnc named[1034]: reloading zones succeeded
Jul 17 01:01:31 frank-cnc named[1034]: all zones loaded
Jul 17 01:01:31 frank-cnc named[1034]: running
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------
Jul 17 05:22:45 frank-cnc named[1065]: zone frankfrank.org/IN: Transfer started.
Jul 17 05:22:45 frank-cnc named[1065]: transfer of 'frankfrank.org/IN' from 192.168.1.97#53:
connected using 192.168.65.133#49694
Jul 17 05:22:45 frank-cnc named[1065]: zone frankfrank.org/IN: transferred serial 201607111
Jul 17 05:22:46 frank-cnc named[1065]: transfer of 'frankfrank.org/IN' from 192.168.1.97#53:
Transfer completed: 1 messages, 28 records, 5602 bytes, 0.010 secs (560200 bytes/sec)
Jul 17 05:22:45 frank-cnc named[1065]: zone frankfrank.org/IN: sending notifies (serial
201607174)
------------------------------------------------------------------------------------------------
------------------------------------------------------------------------
```

-3-

Figure C.3.: System Log 3 for Slave Nameserver